Methods of privacy preservation

David Otter
Betreuer: Marcel von Maltitz
Innovative Internet-Technologien und Mobilkommunikation SS2015
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: otter@in.tum.de

KURZFASSUNG

Die Einhaltung von gesetzlichen wie ethischen Vorgaben bei der Gewährleistung von Privatsphäre der Nutzer stellt Softwareentwickler vor große Herausforderungen. Diese Arbeit beschäftigt sich mit der Frage, wie Privatsphäre von Seiten der Entwickler einer Software schon bei deren entstehen systematisch berücksichtigt werden kann. Dafür werden Software Pattern gesammelt, klassifiziert und erläutert wie diese, verschiedene Aspekte der Privatsphäre positiv beeinflussen.

Schlüsselworte

Software Pattern, Privacy Pattern, Privacy Enhancement

1. EINFÜHRUNG

1.1 Motivation

Der Schutz der Privatsphäre ist ein wichtiges Thema, welches überall dort von besonderer Bedeutung ist, wo große Mengen an sensitiven Daten entstehen. Das Thema ist auch in letzter Zeit wieder vermehrt in den Fokus der Öffentlichkeit gerückt und außerdem gibt es bereits sehr umfangreiche Literaturbestände zur Bedeutung von Privacy. Vor einem Problem hingegen stehen solche Softwareingenieure, welche darauf achten möchten oder müssen, dass sie Privacy by Design ermöglichen. Der Begriff Privacy by Design wurde dabei erstmals in einem Report [22] 1995 über Privacy Enhancing Technologies erwähnt und beschreibt einen Ansatz, bei dem Privatsphäre über den gesamten Lebenszyklus einer Software, mit der Softwarearchitektur beginnend und über alle Organisationsstufen hinweg, berücksichtigt wird.
Typischerweise wird in der Informatik versucht, wiederkehrende Probleme durch erprobte Muster zu lösen und Privacy ist mittlerweile zu solch einer wiederkehrenden Problemstellung geworden. Solche Design Patterns haben in der Informatik lange Tradition, weshalb es auch hierzu viele Sammlungen an solchen und Literatur dazu im Allgemeinen gibt. Bei der Suche nach Privacy Design Pattern fällt auf, dass insbesondere übersichtliche Sammlungen solcher, fehlen. Außerdem verwendet jede Arbeit ihr eigenes Schema, nach welchem die Pattern beschrieben werden und es gibt keine Arbeit bei der die Pattern nach den Design Strategies von Hoepman [1] eingeteilt werden. Dabei hat Hoepmam seine Design Strategies nach bestehendem ISO 29100 Privacy framework [28] ausgewählt, welche es erleichtern das richtige Muster für den richtigen Anwendungszweck zu finden. Deshalb sollen in dieser Arbeit verschiedene Quellen, welche sich bereits mit einzelnen Pattern beschäftigt haben gesammelt, die Pattern nach einem einheitlichen Schema beschrieben und nach angewendeter Strategie kategorisiert werden.

1.2 Privacy

Eine allgemein anerkannte Definition für Privacy zu finden ist auf Grund der Vielzahl an Aspekten, welches von diesem Konzept betroffen ist, schwierig. David H. Flaherty [12] nennt 13 Aspekte von Privacy, welche so in seiner Arbeit angeführt und auch häufig in der Literatur zitiert werden:

1. The right to individual autonomy
2. The right to be left alone
3. The right to a private life
4. **The right to control information about oneself**
5. **The right to limit accessibility**
6. The right of exclusive control of access to private realms
7. The right to minimize intrusiveness
8. **The right to expect confidentiality**
9. The right to enjoy solitude
10. The right to enjoy intimacy
11. **The right to enjoy anonymity**
12. The right to enjoy reserve
13. **The right to secrecy**

Einige dieser Aspekte, welche hier hervorgehoben wurden, können direkt auf Software angewendet werden. Wichtig an dieser Stelle ist, dass Sicherheit, gerne auch als Synonym für Privacy verwendet wird. Informationssicherheit kann über das weit verbreitete **C**onfidentiality, **I**ntegrity, and **A**vailability Triaden Modell beschrieben werden. Nun kann die Sicherheit der eigenen Daten aber nicht verhindern, dass diese Daten absichtlich an dritte Parteien weiterverkauft oder für Marketingzwecke ein komplettes, detailliertes Profil von einem selbst erstellt wird. Auf der anderen Seite kann eine hochgradig unsichere Anwendung, welche keine persönlichen Daten sammelt, nicht die Privatsphäre verletzen. Deshalb wird Sicherheit nicht im Fokus dieser Arbeit stehen.
Jaap-Henk Hoepman beschäftigt sich in seiner Forschung insbesondere mit Privacy by Design und hat auch eine Vielzahl an paper darüber verfasst. Seine Arbeit, welche auch als Grundlage der Struktur der restlichen Arbeit dient, hat es geschafft eine fundierte Ordnung in das Thema zu bringen.

1.3 Nomenklatur

Der Begriff des Pattern wurde ursprünglich von Christopher Alexander (1979) geprägt: „A pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution“. In der Softwaretechnik werden solche Entwurfsmuster für wiederkehrende Probleme in ver-

doi: 10.2313/NET-2015-09-1_10

schiedene Abstraktionsebenen unterteilt. Das Hauptwerk zu Design Patterns in der Softwaretechnik wurde dabei von der Gang of Four verfasst [29]. Das Namensschema und die Grenzen zwischen den Kategorien wird nach Hoepman [1] übernommen.
Die höchste Abstraktionsstufe Design Strategies, auch bekannt als Architeture Patterns, sind für die fundamentale strukturelle Organisation von Softwaresystemen verantwortlich. Design Patterns, wie sie nachfolgend gesammelt sind, sind Muster welche ein Subsystem, also einzelne Komponenten beschreiben. Privacy Enhancing Technologies (PETs) sind hingegen konkrete Ausprägungen eines Design Patterns, welches einen positiven Einfluss auf die Privatsphäre besitzt. Um den Unterschied zwischen einem Design Pattern und einer PET klarer zu machen, eignet sich besonders ein Beispiel von Hoepman: Idemix, ein kryptografisches Protokoll für die Privatsphäre erhaltenden Authentifizierung und den Transfer von Zertifikatattributen [23], ist die Ausarbeitung einer konkreten Technologie auf Basis des Pattern Anonymous Credentials. Aber nicht immer ist eine so klare Abgrenzung möglich.

2. DESIGN-PATTERN-VERZEICHNIS

Für die Strukturierung der Design Patterns werden die 8 Privacy Design Strategies, welche von Hoepman definiert wurden, verwendet [1]. Die Privacy Pattern wiederum werden nach folgendem Schema beschrieben:

1. Name mit Quelle
2. Kontext
3. Problem
4. Lösung
5. Beispiele
6. Ähnliche Ansätze und Querverweise

2.1 Minimise

Die einfachste Art, Privatsphäre zu gewährleisten, ist die Anzahl an gesammelten Daten zu minimieren. Ziel der Strategie ist, nur noch die für die Qualität des angebotenen Diensts notwendigen Daten zu sammeln.

2.1.1 Strip Invisible Metadata [3]

Bei der Verwendung von nutzer-generierten Dateien werden oft unsichtbare Metadaten mitgeliefert, welche vom Service nicht benötigt werden.
Die meisten Nutzer sind sich dieser Informationen, da versteckt, nicht bewusst und haben auch kein Wissen darüber, wie sie diese Metadaten entfernen können.
Solange dem Nutzer Metadaten nicht sichtbar gemacht werden können, oder diese für den Service überhaupt nicht benötigt werden, sollen sie gelöscht werden, womit die Anforderung an *Minimise* erfüllt wird.
Als Beispiel für diesen Ansatz darf Twitter gesehen werden. Beim Hochladen von Fotos auf deren Plattform werden alle Exif-Daten, ein Standard für Metainformationen in der Fotografie, von den Fotos entfernt. [4]

2.1.2 Location Granularity [13]

Viele Dienste benötigen Standortdaten, um sie selbst zu bearbeiten oder an andere Dienste weiterzuleiten.
Solche Daten werden oft durchgehend und in maximaler zur Verfügung stehender Genauigkeit gesammelt und beeinträchtigen dadurch die Privatsphäre des Nutzers.
Die Auflösung, in der Standortdaten gesammelt werden, ist oftmals für den angebotenen Service gar nicht nötig und kann deshalb heruntergesetzt werden. Damit werden die gesammelten Daten auf ein Minimum reduziert, ohne die Qualität des angebotenen Dienstes zu beeinträchtigen.
Ein Beispiel, wo dieses Pattern angewendet werden kann, ist ein Wetterdienst. Auch mit einem genauen GPS-Signal kann keine genauere Wettervorhersage geliefert werden, als mit der Postleitzahl.
Wenn die Wahl über den gewählten Detailgrad der Standortdaten von dem Nutzer selbst getroffen werden kann, wird damit auch das Pattern der *Blurred Personal Data* (2.6.4) und damit die *Control*-Strategie erfüllt.

2.2 Hide

Keine gesammelten oder übertragenen Daten sollen als Klartext vorliegen. Je nach Kontext sollen die Daten vor allen versteckt werden, um keine Musteranalyse zuzulassen, oder vor Dritten, um Vertraulichkeit gewährleisten zu können.

2.2.1 Constant Length padding [2]

In Netzwerken wird Verschlüsselung eingesetzt, um die darüber verbreiteten Daten zu schützen. Trotzdem können immer noch Metadaten, wie die Länge der Nachricht eingesehen werden.
Diese Metadaten können einerseits die Wirksamkeit der Verschlüsselung kompromittieren. Gerade in Netzwerken, wo symmetrische und asymmetrische Verschlüsselung kombiniert werden, ist dies ein Problem, da Nachrichten an jedem Knotenpunkt entschlüsselt werden und sich bei jeder Entschlüsselung die Nachrichtenlänge verringert. Andererseits gehört die Länge auch zu den sensitive Daten, welche geschützt werden sollten.
Deshalb werden alle Paketlängen uniformiert und falls notwendig, um die vorgegebene Länge zu erreichen, die Pakete mit einem Padding aufgefüllt. Die Länge als Teil der Metadaten ist also nicht mehr einfach von außen einsehbar.
Ein Beispiel für die Verwendung von *Constant Length padding* findet sich in Kombination mit der *Layered encryption*, auch bekannt als *Onion Routing* und noch anderen Techniken in Mix Networks. Das anonyme Netzwerk Tor baut auf diesen Prinzipien auf.
Eine weiteres Pattern welches sich mit Metadaten beschäftigt ist *Strip Invisible Metadata*, welches bei impliziten wie der Länge aber nicht angewendet werden kann.

2.2.2 Cover Traffic [2]

In Anonymen Netzwerken wird versucht, sich gegen alle mögliche Arten von Angriffen und Traffic Analysen zu schützen.
Insbesondere Netzwerke mit geringer Latenz können auf Grund dieser Echtzeitanforderungen leichter analysiert werden, aber auch andere Mix Networks sind bestrebt die Sicherheit weiter zu erhöhen. Eine Möglichkeit, solch eine Attacke auf ein low-latency Netzwerk durchzuführen ist, indem Bandbreiteschwankungen induziert werden und dann beobachtet wird, wie diese durch das Netzwerk sickern. [24]
Durch Dummy Pakete wird der Versuch, einzelne Pakete oder Paketströme nachzuvollziehen, erschwert, oder die Art des verwendeten Protokolls maskiert. Unter anderem können Dummy Nachrichten in einem Mix Network verschickt

werden, welche bei zufälligen Knotenpunkten enden und damit die Identifizierung des Empfängers erschweren.
Ein ähnliches Pattern kann auch für Datenbanken verwendet werden. Dabei wird in Ergebnissen der Antwort von (No-)SQL Anfragen die Reihenfolge der erhaltenen Daten vertauscht, oder die Ergebnisse werden verfälscht, indem einzelne Dummy Datensätze eingefügt werden.
Cover Traffic wird unter anderem von Tarzan [14] verwendet, einem peer-to-peer-Anonymisierungsdienst.

2.2.3 Layered Encryption [2]

In Mix Networks soll Verschlüsselung eingesetzt werden, um die Daten, die darin versendet werden zu schützen.
Bei einem einfachen Verschlüsselungsansatz reicht es aus, einen einzigen Knotenpunkt zu kompromittieren, um den gesamten Verbreitungsweg nachvollziehen zu können und damit Sender und Empfänger zu identifizieren.
Die Lösung besteht in einer schichtweisen Verschlüsselung, bei der jeder Knotenpunkt nur den unmittelbar nächsten Nachbarn kennt. Als Vorbereitung auf das Versenden eines Pakets wird also zuerst eine Route festgelegt. Neben der End-to-End-Verschlüsselung haben alle Knotenpunkte und deren Nachbar ihren separaten symmetrischen Schlüssel und die Nachricht wird auf dem Weg schichtweise entschlüsselt. Die Daten sind also von außen und bis auf die nötigen Informationen über den nächsten Knotenpunkt auch innerhalb des Netzwerks geschützt.
Synonym zur *Layered Encryption* wird auch von *Onion Routing* gesprochen. Die populärste Anwendung davon, welche auch auf *Constant Length Padding* zurückgreift, ist das Netzwerk Tor.
Das Pattern erfüllt auch die Voraussetzungen der Strategie *Separate*, da die Informationen über den zurückgelegten Weg im Netzwerk in Einzelschritte zerlegt werden.

2.3 Separate

Speicherung, Transport und Verarbeitung von Daten soll möglichst verteilt sein. Besonders die Verknüpfung verschiedener Daten soll damit erschwert werden.

2.3.1 CP-ABE [25]

In verteilten Systemen sollen Daten nur von bestimmten Personen einsehbar sein. Die Gruppe der Personen kann über bestimmte Anmeldedaten oder andere Attribute festgelegt werden. Solch eine Richtlinie kann über einen vertrauenswürdigen Server kontrolliert werden, welcher die Daten speichert und den Zugang regelt.
Das Problem besteht nun darin, die Vertrauenswürdigkeit der Daten zu gewährleisten, selbst wenn ein Server, welcher die Daten gespeichert hat, nicht vertrauenswürdig ist. Eine bewährte Lösung besteht in Verschlüsselung, wobei die meisten eingesetzten public key Verfahren keine Möglichkeit bieten, einzelne Nutzerrollen umzusetzen und zu verwalten.
Bei Ciphertext-Policy Attribute-Based Encryption wird jeder private key eines Nutzers mit einer bestimmten Anzahl an Attributen verknüpft und die Zugriffsstruktur in der verschlüsselten Datei hinterlegt. Die Zugriffsstruktur wird dabei als eine Baumstruktur umgesetzt, in welcher ein Knoten, eine UND Bedingung, ODER Bedingung oder ein Attribut ist. Nur wenn dieser Zugriffsbaum durch die mit dem private key verknüpften Attribute erfüllt wird, ist dem Nutzer die Entschlüsselung der Daten möglich.
Von einem der Autoren von [25] wurde ein Toolkit zur Umsetzung von *CP-ABE* erstellt, welches jedoch vom Autor selbst ausdrücklich nicht für den Produktiveinsatz in sicheren Systemen empfohlen wird. [26]
Neben der sicheren verteilten Speicherung von Daten ermöglicht das Pattern auch eine umfangreiche Zugriffskontrolle. Da diese mathematisch im Pattern verankert ist und nicht einer Instanz vertraut werden muss, ist damit auch die *Enforce*-Strategie erfüllt wird.

2.3.2 Domain Specific Pseudonyms

Anbieter vieler verschiedener Dienste wie Google oder aber auch die Öffentliche Hand besitzen die Möglichkeit, Daten aus verschiedensten Lebensbereichen der Nutzer zu sammeln.
Durch Verknüpfung der erworbenen Daten bietet sich ein umfangreiches Bild des Nutzers, welches weit über jenes hinausgeht, welches sich bei der Nutzung nur eines Dienstes ergeben würde. Selbst bei der Verwendung eines Pseudonyms kann die Eindeutigkeit der Daten dazu ausreichen, ein einzelnes Individuum zu identifizieren.
Eine Lösung besteht darin, für jeden angebotenen Dienst ein eigenes Pseudonym zu erstellen, um dadurch die Verlinkung der Informationen zu erschweren. Dieses Vorgehen stellt eine logische Separation der Daten dar.
Dieser Ansatz wurde auch für die Deutsche Identitätskarte gewählt [10], wo jedem Dienstleister, welche diese Art der Authentifizierung verwendet, ein eigenes Pseudonym zugewiesen wird. Damit können selbst bei Verlust der Daten keine Informationen über mehrere Dienstleister hinweg kombiniert werden.
Pseudonyme im Allgemeinen fallen unter die Strategie des Information *Hiding*, welche auch bei *Domain Specific Pseudonyms* erfüllt wird.

2.4 Aggregation

Informationen sollen maximal aggregiert, also der Detailgrad minimiert werden, so dass die Qualität des Service immer noch erfüllt werden kann. Dadurch kann die Anzahl und der Detailgrad von Daten noch nach dem Sammeln verringert werden, auch wenn sie ursprünglich in dieser Auflösung gebraucht wurden. Zur Aggregation können dabei beliebige Funktionen verwendet werden, nach deren Anwendung der Detailgrad der gespeicherten Daten geringer ist als vorher.

2.4.1 Aggregation over time or space

Daten werden oft sehr detailliert erhoben und in diesem Detailgrad unter anderem für Echtzeitanwendungen verwendet und abgespeichert.
Mit diesen Daten lassen sich zum Teil Tagesabläufe und andere höchst persönliche Einsichten in das Leben von anderen Personen generieren.
Bei der Aggregation over time können alte Daten oft ohne Einfluss auf die Qualität des angebotenen Service aggregiert werden, da zum Beispiel bloß die Summe für statistische Analysen und Auswertungen benötigt wird. Bei der Aggregation over space werden mehrere Datenquellen, wie Sensoren in Clustern zusammengefasst. Die gesammelten Daten eines Clusters werden dann aggregiert, bevor sie zur Verarbeitung weitergeschickt werden. Damit können auch in Echtzeit benötigte Daten geschützt werden.

doi: 10.2313/NET-2015-09-1_10

Ein Beispiel für die Anwendung von Aggregation findet sich in [6], wo die Möglichkeiten analysiert wurden, ein intelligentes Stromnetz unter Berücksichtigung der Privatsphäre zu installieren. Als weitere Beispiele werden in [5] der Gesundheitssektor, Cloud Services, allgemeine Sensornetzwerke oder die Überwachung von Menschen beschrieben.

2.5 Inform

Nutzer sollen maximale Transparenz über Sammlung, Verarbeitung und Speicherung von Daten erhalten. Auch Informationen über Maßnahmen, welche zum Schutz der Daten getroffen wurden, fallen unter die *Inform*-Strategie.

2.5.1 Asynchronous notice [17]

Viele mobile Geräte zeichnen Standortdaten oder andere sensorbasierte Werte auf und zwar auf eine für den Nutzer nicht transparente Art und Weise.
Selbst wenn die Genehmigung dafür vom Nutzer erteilt wurde, kann oft nach langer Zeit nicht mehr garantiert werden, dass dieser sich dessen noch bewusst ist. Eine andere Möglichkeit ist, dass diese Genehmigung von einem vorherigen Besitzer des Gerätes erteilt wurde.
Um größtmögliche Transparenz zu erreichen, sollten nach einem vorher definierten Muster asynchrone Erinnerungsnachrichten mit der Aufklärung über aktuell erhobene Daten verschickt werden. Asynchron beschreibt in diesem Zusammenhang die Eigenschaft, dass die Nachricht nicht nur bei der erstmaligen Verwendung des Sensors geschickt wird.
Der mittlerweile eingestellte Dienst von Google Latitude bot eine Möglichkeit sich mit einer Email informieren zu lassen, wann immer der Standort mit anderen externen oder internen Anwendungen geteilt wurde.

2.5.2 Privacy dashboard [27]

Die Menge an Daten, welche von mobilen Anwendungen oder anderen Services gesammelt werden, sind sehr umfangreich und unübersichtlich. Eine Datenschutzerklärung, in welcher über alle relevanten Eingriffe in die Privatsphäre aufgeklärt wird, hat für Nutzer in der Praxis wenig Relevanz, da diese mit juristischen Floskeln gefüllt wird und nicht mehr die Aufgabe erfüllen die Nutzer bei der Abwägung von Vor- und Nachteilen eines Dienstes zu unterstützen.
Gesucht ist also eine Lösung, mit der über alle gesammelten Daten übersichtlich und leicht auffindbar informiert wird, damit der Nutzer nicht erst im Nachhinein herausfindet, welche Daten von ihm gesammelt und verarbeitet wurden.
Die Lösung dafür ist, alle relevanten Daten auf einer Seite zu sammeln und soweit zu komprimieren, dass der Nutzer die Übersicht behält, ihm aber trotzdem noch klar ist, welche Daten von ihm in welchen Bereichen erhoben werden. Dieses *Privacy Dashboard* muss auch noch für den Nutzer leicht auffindbar sein, wenn er danach sucht.
Beispiel für den Einsatz des Patterns ist bei Android die Übersicht über die von Apps eingeforderten Berechtigungen, welche vor der Installation gezeigt werden, aber auch noch im Nachhinein jederzeit eingesehen werden können.
Wenn über das *Privacy Dashboard* dem Nutzer auch noch Kontrollmöglichkeiten in die Hand gegeben werden, wird damit auch die *Control*-Strategie erfüllt. Beispiel dafür ist die mittlerweile bestätigte Möglichkeit in Android M, ähnlich dem bereits existierenden System von Cyanogen Mod, Apps einzelne Berechtigungen auch wieder entziehen zu können.

2.5.3 P3P [18]

P3P war der Versuch, einen maschinell auswertbaren Standard zu schaffen, mit dem Nutzer über von ihnen gesammelte Daten informiert werden können.
Das Programm war Gegenstand einiger wissenschaftlicher Arbeiten, wurde aber mit Version 1.1 eingestellt. Neben der mangelnden Akzeptanz, besonders von Seiten der großen Browser-Hersteller, gab es Probleme die Einhaltung der Versprochenen Datenschutzversprechen zu kontrollieren. Da die P3P kompatiblen Datenschutzerklärungen von den Services selbst erstellt wurden, gab es keine Möglichkeit sicherzustellen, dass diese auch eingehalten wurden.

2.6 Control

Als Gegenstück zur *Inform*-Strategie müssen dem Nutzer auch Möglichkeiten gegeben werden, um Kontrolle über die eigenen Daten zu erhalten. Mit eingeschlossen ist auch die Einfachheit und Zugänglichkeit der Werkzeuge, die dafür zur Verfügung gestellt werden.

2.6.1 Encryption with user controlled key [20]

Einerseits wollen Nutzer den Komfort von online-basierten Angeboten nutzen, andererseits aber trotzdem die Kontrolle über ihre Daten behalten. Eine Möglichkeit der Kontrolle ist symmetrische Verschlüsselung der abgespeicherten Daten mithilfe eines kryptografischen Keys.
Solange der Anbieter, welcher die Daten speichert, auch im Besitz des Keys ist, kann keine vollständige nutzer-basierte Kontrolle garantiert werden. Entweder der Anbieter selbst, oder dritte Parteien, welche Serviceleistungen für den Anbieter durchführen, können diese Dateien einsehen. Auch der Zugriff von Regierungsorganisationen kann im Allgemeinen nicht ausgeschlossen werden.
Zur Lösung dieses Problems können Dateien vor dem Hochladen verschlüsselt und der Schlüssel lokal gespeichert werden. Auch eine Lösung, bei dem der Schlüssel online gespeichert wird, aber die Passphrase lokal einzugeben ist, ist als Kompromiss zwischen Benutzerfreundlichkeit und nutzerbasierter Kontrolle möglich.
Durch die Trennung von Daten und dem zugehören Schlüssel wird auch die *Separate*-Strategie erfüllt. Eine Beispielanwendung, welche zwar den Schlüssel physikalisch trennt, aber nicht ganz die Anforderung des user-controlled Keys erfüllt, ist KeyNexus [21]. Dabei handelt es sich um einen cloud-Service ausschließlich für Verschlüsselungs-Keys.

2.6.2 Token based access

Viele Anwendungen bieten den Zugriff durch Dritte über eine vordefinierte API, genauso wie viele Anwendungen Daten für den Nutzer speichern.
Sowohl für die API als auch für persönliche Daten soll es eine für den Nutzer möglichst einfache, aber dennoch sichere Möglichkeit geben, festzulegen, welchen Anwendungen oder andere Personen er Zugriff gibt.
Solch ein Zugriff wird üblicherweise mithilfe von Token realisiert, welche die zugreifende Partei gegenüber dem Dienst autorisiert. Wenn individuelle Token vergeben werden, gibt es für den Nutzer auch eine einfache Möglichkeit, Zugriffe zu identifizieren und einzelnen Token die Zugriffsrechte wieder

doi: 10.2313/NET-2015-09-1_10

zu entziehen.
Ein Beispiel für die Verwendung von individuellen Token ist tum online, wo bereits erteilte Genehmigungen auch wieder zurückgenommen werden können.
Bei google drive hingegen können Daten mithilfe eines privaten links geteilt werden. Auch hier können erteilte Genehmigungen wieder zurückgenommen werden, allerdings nur indem der private link, für alle die ihn besitzen, deaktiviert wird.

2.6.3 Broadcasting with client side selecting

Um ortssensitive Services, wie zum Beispiel Wetterberichte oder Gastronomieempfehlungen anbieten zu können, müssen vorher entsprechende GPS-Daten gesammelt und an den Server gesendet werden.
Jede Sammlung von Daten und das Versenden dieser hat Einfluss auf die Privatsphäre, insbesondere gibt der Nutzer in diesem Moment die Kontrolle über seine eigenen Daten ab.
Eine Möglichkeit ist die Rasterung in große Gebiete, welche zum Beispiel auf Bundesländerebene geschehen könnte. Daraufhin werden an alle in diesem Gebiet befindlichen Empfänger ein Broadcast mit allen für dieses Raster relevanten Daten gesendet. Der Nutzer hat dann die Möglichkeit aus den empfangenen Informationen, die für ihn relevanten, auszuwählen.
Für das Wetterberichtbeispiel würde das bedeuten, dass der Nutzer alle Daten von dem Bundesland in dem er sich gerade befindet, empfängt. Aus diesen Daten kann der Nutzer dann lokal das Wetter, für den Ort an dem er sich gerade befindet, auswählen.
Neben der zusätzlichen Kontrolle wird auch die Menge an gesammelten Daten verringert, weshalb damit auch die *Minimise* Strategie erfüllt wird.

2.6.4 Blurred Personal Data [19]

Viele Websites und andere Dienste benötigen Informationen wie den Standort des Nutzers.
Nutzer sollten die Kontrolle über den Detailgrad der von ihnen übermittelten Daten erhalten.
Durch eine Auswahlliste über die der Nutzer selbstständig den Detailgrad der übermittelten Daten bestimmen kann, wird die Kontrolle über die eigenen Daten erhöht. Eine solche Lösung könnte entweder individuell eingepflegt werden oder als Standard für eine komplette Plattform realisiert werden. Bei zweiterem hätte der Nutzer dann an jeder Stelle, an der er aufgefordert wird Standortdaten preiszugeben, die Möglichkeit, den Detailgrad selbst zu bestimmen.
In der zitierten Literatur wird der Mockup eines Browser-Addons gezeigt, über den dann der Nutzer die Auswahlmöglichkeit zwischen den Standort der Stadt, der Postleitzahl oder des genauen Standort hätte. Auch eine absolut festgelegte Unschärfe ist vorstellbar, hier muss aber bedacht werden, dass bei mehrfacher Standortabfrage die Streuung mathematisch stark reduziert werden kann.
Ein ähnlicher Ansatz bei dem sich der Entwickler aber auf eine einzige sinnvolle Auflösung im Vorhinein festlegen muss, ohne dem Nutzer eine Wahl zu bieten, wird durch *Location Granularity* als Pattern der *Minimise*-Strategie verfolgt.

2.7 Enforce

Das Vorhandensein einer Datenschutzerklärung, welche auch alle gesetzlichen Vorgaben erfüllt, ist eine Strategie um die Privatsphäre zu erhöhen. Allerdings nur dann wenn die Einhaltung der Datenschutzerklärung auch durchgesetzt wird.

2.7.1 Sticky Policies [8]

Gesammelte Daten werden oft zwischen verschiedenen internen und externen Parteien ausgetauscht. Um dabei die Privatsphäre der Nutzer zu wahren, gibt es eine Vielzahl an Verträgen, service-level agreements, spezielle frameworks und Auditing-Verfahren.
Dabei hat der Nutzer allerdings keine individuelle Kontrolle mehr, wie mit seinen Daten umgegangen werden soll.
Sogenannte Sticky Policies sind Metadaten, welche den persönlichen Daten angehängt und dann auch mit den Daten so an andere Parteien weitergegeben werden. Sie bestimmen die Voraussetzungen und Grenzen, in welchen die Daten verarbeitet werden dürfen.
Im Zuge des EnCoRe-Projekts [9] wurde eine Architektur geschaffen bei der eine Trust Authority die Einhaltung der Policies der Parteien prüft und erst dann den Zugang zu den verschlüsselten Daten gewährt.

2.7.2 k-Anonymity [15]

Das Konzept der k-Anonymity wurde ursprünglich für den akademischen Einsatz erdacht. Als Kontext dienen dabei Datenbanken von Instituten, wie öffentliche Einrichtungen oder Banken, welche ihre Daten für Forschungszwecke teilen möchten.
Das Problem besteht darin, eine Version dieser Daten zu veröffentlichen, von denen garantiert ist, dass sie die Anonymität der Individuen gewährleisten. Es darf nicht möglich sein, dass einzelne Personen sich identifizieren lassen. Über relinking könnten kritische Daten wie die Gesundheitsakte mit öffentlich zugänglichen Daten verknüpft werden, um dadurch ein Gesamtprofil zu erhalten. Schon einige wenige Daten können dabei in Kombination eindeutig sein.
Bei der Verwendung von k-Anonymity werden die Daten in Äquivalenzklassen eingeteilt, wobei jede dieser Klassen mindestens k Einträge beinhaltet. Von diesen Daten werden dann alle Identifikatoren entfernt. Als Quasi-Identifikatoren bezeichnet man dabei Tupel an Attributen, welche ein Individuum eindeutig identifizieren können. Deshalb werden die Äquivalenzklassen so gewählt, dass alle Datensätze einer Klasse die gleichen Quasi-Identifikatoren besitzen. Darunter fallen auch alle Attribute, welche mithilfe von anderen öffentlich zugänglichen Datenbanken, zu Quasi-Identifikatoren werden. Zur Bildung der Äquivalenzklassen muss teilweise die Auflösung der Daten verringert werden. Ein Beispiel für Identifikatoren ist der Name und für Quasi-Identifikatoren die Attribute des Tupels (Postleitzahl, Geburtsdatum, Alter).
Das Verfahren wurde im Jahr 2002 patentiert [16].
Zur eigentlichen Anonymisierung, also der Erstellung der Äquivalenzklassen, können verschiedene Methoden verwendet werden, welche sich typischerweise der *Aggregate*-Strategie bedienen.

2.8 Demonstrate

Die *Demonstrate*-Strategie ist der nächste logische Schritt nach *Enforce*. Die Strategie legt fest, dass die Einhaltung der Datenschutzerklärung jederzeit und nachvollziehbar demonstriert werden kann. Wenn davon ausgegangen wird, dass die Umsetzung der *Inform*-Strategie ehrlich erfolgt und die

doi: 10.2313/NET-2015-09-1_10

Sammlung und Verarbeitung rechtskonform ist, kann auch jegliches Pattern, welches über die gesammelten Daten informiert, die *Demonstrate*-Strategie erfüllen.

2.8.1 Accounting

Es kommt vor, dass Datenlecks in Firmen geleugnet werden bis es nicht mehr widerlegbare Beweise dafür gibt. Oder die Lecks werden erst entdeckt, wenn die Daten bereits im Umlauf sind.
Neben der Zugriffskontrolle gibt es kaum Möglichkeiten, im Nachhinein zu zeigen, wann von wem auf welche Daten zugegriffen wurde, oder von welchem Dienst die Daten verarbeitet wurden. Das führt zu einer Grundskepsis von Nutzern gegenüber online basierten Diensten.
Um die Einhaltung der Datenschutzrichtlinien jederzeit nachweisen zu können, sollte ein System auf Servern, welche persönliche Daten speichern, eingerichtet werden, welche alle relevanten Zugriffe auf die Daten und darauf ausgeführte Funktionen, lückenlos aufzeichnet. Das System muss zuverlässig und fehlerresistent sein und sollte einen Output ähnlich einem Log erzeugen, welcher in geeigneter Art und Weise einsehbar sein muss. Damit sollen alle externen und internen Zugriffe aufgezeigt werden.

2.8.2 Auditing

Viele Firmen versuchen zu beteuern, dass ihre Software sicher ist und auch alle Datenschutzrichtlinien eingehalten werden. Diese Aussagen sind nicht nachprüfbar, weil viel Code closed source ist. Aber auch bei open source Projekten gehen die Lines of Code schnell in die hunderttausend. Vielen Nutzern fehlt die Expertise oder aber auch die Zeit, diesen Code zu prüfen.
Eine mögliche Lösung ist Software-*Auditing* von unabhängigen Experten, welche den Code auf Backdoors oder unbeabsichtigte Schwachstellen prüfen.
Ein Beispiel, wo mit Hilfe von *Auditing* [7] versucht wurde, das Vertrauen der Nutzerbasis wiederherzustellen ist TrueCrypt, eine Verschlüsselungssoftware deren Entwicklung Mai 2014 eingestellt wurde, weil es angeblich schwerwiegende Sicherheitslücken darin gebe.
Im Sinne der *Inform*-Strategie könnte *Auditing* standardisiert und nach erfolgreicher Prüfung der Software eine Art Qualitätssiegel, ähnlich der *trusted shops* Auszeichnung im online shopping Bereich, vergeben werden.

3. CONCLUSIO UND AUSBLICK

Die Arbeit zeigt, dass es bereits zu allen Strategien, welche die Privatsphäre erhöhen, Design Pattern gibt mit welchen Privacy by Design realisiert werden kann. Viele Pattern setzen gleich auf mehreren Ebenen an und werden in einzelnen Projekten schon erfolgreich eingesetzt. Indem das Bewusstsein für und die Verbreitung von Privacy Pattern erhöht wird, kann ein wichtiger Beitrag zum Schutz der Privatsphäre und damit das Vertrauen der Nutzer in solche Software geschaffen werden.
Nichtsdestotrotz benötigt der Ansatz, den Schutz der Privatsphäre bereits auf der Systemdesignebene zu gewährleisten, sicherlich noch weitere Bemühungen in der Forschung. Projekte wie PRIPARE [11] könnten dafür eine entscheidende Rolle spielen.
Als Ausblick sollte die erfolgte Sammlung an Pattern in Zukunft noch erweitert und mit weiteren Detailinformationen wie zur Implementation bereichert werden.

4. LITERATUR

[1] HOEPMAN, Jaap-Henk. Privacy Design Strategies. 2012.

[2] HAFIZ, Munawar. A pattern language for developing privacy enhancing technologies. Software: Practice and Experience, 2013, 43. Jg., Nr. 7, S. 769-787.

[3] privacypatterns.org - Strip invisible Metadata, http://privacypatterns.org/patterns/Strip-invisible-metadata, accessed: 28-05-2015

[4] twitter - faq, https://support.twitter.com/articles/20169321-posten-von-fotos-auf-twitter, accessed: 28-05-2015

[5] SHI, Elaine, et al. Privacy-Preserving Aggregation of Time-Series Data. In: NDSS. 2011. S. 3.

[6] KURSAWE, Klaus; DANEZIS, George; KOHLWEISS, Markulf. Privacy-friendly aggregation for the smart-grid. In: Privacy Enhancing Technologies. Springer Berlin Heidelberg, 2011. S. 175-191.

[7] Open Crypto Audit Project, https://opencryptoaudit.org/, accessed: 14-05-2015

[8] PEARSON, Siani; MONT, Marco Casassa. Sticky policies: an approach for managing privacy across multiple parties. Computer, 2011, 44. Jg., Nr. 9, S. 60-68.

[9] MONT, Marco Casassa; SHARMA, Vaibhav; PEARSON, Siani. EnCoRe: dynamic consent, policy enforcement and accountable information sharing within and across organisations. HP Laboratories technical Report: HPL-2012-36, 2012.

[10] BENDER, Jens, et al. Domain-specific pseudonymous signatures for the german identity card. In: Information Security. Springer Berlin Heidelberg, 2012. S. 104-119.

[11] PReparing Industry to Privacy-by-design by supporting its Application in REsearch, http://pripareproject.eu/, accessed: 10-05-2015

[12] David H. Flaherty. Protecting Privacy in Surveillance Societies: The Federal Republic of Germany, Sweden, France, Canada, and the United States. University of North Carolina Press, Chapel Hill, NC, USA, 1989.

[13] privacypatterns.org - Location Granularity, http://privacypatterns.org/patterns/Location-granularity, accessed: 27-05-2015

[14] FREEDMAN, Michael J.; MORRIS, Robert. Tarzan: A peer-to-peer anonymizing network layer. In: Proceedings of the 9th ACM conference on Computer and communications security. ACM, 2002. S. 193-206.

[15] SWEENEY, Latanya. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2002, 10. Jg., Nr. 05, S. 557-570.

[16] Systems and methods for deidentifying entries in a data source - https://www.google.com/patents/US7269578, accessed: 01-06-2015

[17] privacypatterns.org - Asynchronous Notice, http://privacypatterns.org/patterns/Asynchronous-notice, accessed:

07-05-2015
[18] P3P, http://www.w3.org/P3P/, accessed: 01-06-2015
[19] CHUNG, Eric S., et al. Development and evaluation of emerging design patterns for ubiquitous computing. In: Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques. ACM, 2004. S. 233-242.
[20] privacypatterns.org - Encryption with user managed keys, http://privacypatterns.org/patterns/Encryption-user-managed-keys, accessed: 01-06-2015
[21] Key Nexus, https://keynexus.net/, accessed: 01-06-2015
[22] Privacy-Enhancing Technologies: The Path to Anonymity. Office of the Information & Privacy Commissioner of Ontario, Registratiekamer, 1995.
[23] Idemix - IBM research zurich, http://www.zurich.ibm.com/idemix/whatitdoes.html, accessed: 06-06-2015
[24] CHAKRAVARTY, Sambuddho; STAVROU, Angelos; KEROMYTIS, Angelos D. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In: Computer Security–ESORICS 2010. Springer Berlin Heidelberg, 2010. S. 249-267.
[25] BETHENCOURT, John; SAHAI, Amit; WATERS, Brent. Ciphertext-policy attribute-based encryption. In: Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007. S. 321-334.
[26] http://hms.isi.jhu.edu/acsc/cpabe/
[27] privacypatterns.org - Privacy Dashboard, http://privacypatterns.org/patterns/Privacy-dashboard, accessed: 06-06-2015
[28] SO/IEC 29100, Information technology – Security techniques – Privacy framework, Technical report, ISO JTC 1/SC 27
[29] GAMMA, Erich, et al. Design patterns: elements of reusable object-oriented software. Pearson Education, 1994.

doi: 10.2313/NET-2015-09-1_10