

hochschule mannheim

Bachelorarbeit

Entwurf einer tragbaren kryptographischen Softwarelösung für den
sicheren Datenaustausch zwischen Unternehmen

eingereicht von: Siyapdje Fabrice Dufils
Matrikelnummer: 1015366
Studiengang: Technische Informatik
Hochschule Mannheim

betreut durch: Prof. Dr. Martin Damm
Hochschule Mannheim

Mannheim, den 30. März 2016

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Es wurden keine anderen als die angegebenen Quellen und Hinweise verwendet.

Die vorliegende Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Mannheim, den 30. März 2016

Siyapdje Fabrice Dufils

Kurzfassung

Die vorliegende Arbeit wurde an der Hochschule Mannheim angefertigt. Dabei sollte ein sicheres Datenaustausch-Programm für Unternehmen entwickelt werden.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Gliederung	1
1.2	Problemstellung und Motivation	1
1.3	Zielsetzung	2
2	Grundlagen und Stand der Technik	3
2.1	Begriffe	3
2.2	Email	7
2.3	Web-upload	8
2.4	File Transfer Protocol (FTP)	9
2.5	Cloud-Service	9
2.6	PGP	10
2.7	X.509	12
2.8	SPKI / SDSI	13
3	Anforderungen	15
3.1	funktionale Anforderungen	16
3.2	Nicht-funktionale Anforderungen [1]	17
3.2.1	Allgemeine nichtfunktionale Anforderungen	17
3.2.2	Wartbarkeit und Erweiterbarkeit	18
3.2.3	Portierbarkeit und Plattform-Unabhängigkeit	18
3.2.4	Daten-und Serverintegrität	18
4	Konzept	19
4.1	Allgemein Architektur	19
4.2	Authentifizierung	19
4.3	Kritische Daten Integrität	22
4.4	Schlüsselaustausch	22
4.5	Server Integrität	22
4.6	Web-Of-Trust (Friends-Konzept)	22
4.7	Übermittlung von Passphrase	24
4.8	Gruppe	24
4.9	Dokumentaustausch	25

4.10	REST (Representational State Transfer)	27
4.11	Integrität der transportierte kritischen Daten (JWS)	29
4.12	JSON Web Token (JWT)	30
4.13	Notar (SGK zurückgewinnen)	30
4.14	Szenarien	31
5	Implementierung und Evalierung	37
5.1	Überblick auf die Technologie	37
5.2	Allgemein Designentscheidungen	37
5.2.1	JSON-Format	37
5.2.2	UTF-8 und Base64	38
5.2.3	HTTP Headers	38
5.3	Frontend	39
5.3.1	Ausstattung von Routes	40
5.4	allgemeine Implementierung Entscheidung von Local-und Remoteserver	40
5.5	LocalServer	40
5.6	Evaluierung	47
5.6.1	Anforderungserfüllung	47
6	Zusammenfassung und Ausblick	50
6.1	Zusammenfassung	50
6.2	Ausblick	52
	Glossar	53
	Literaturverzeichnis	54

Abbildungsverzeichnis

2.1	Asymmetrische Verschlüsselung	4
2.2	Symmetrische Verschlüsselung	5
2.3	exemplarische zertifikat	10
3.1	Allg. funktionale Anforderungen-Überblick	16
4.1	exemplarische Vertrauenbeziehung	23
4.2	Zwischen Benutzer und Group	25
4.3	Public Group zu Public Group	25
4.4	Benutzer-Benutzer Dokumentsaustausch	26
4.5	Benutzer-Benutzer Dokumentsaustausch Ablauf	26
4.6	CryptoOne HTTP Resources nach REST-Spezifikation	28
4.7	JWS Header	29
4.8	Signature Filters	29
4.9	JWT-Token : Erläuterung	30
4.10	Benutzer registrieren	31
4.11	Einloggen	32
4.12	Freund hinzufuegen sequence diagram	32
4.13	Revoke Friend	33
4.14	Datei hochladen sequence diagramm	34
4.15	Neue Gruppe anlegen	36
5.1	Angular: Cookie Konfigurierung	40
5.2	Routes-Übersicht	40
5.3	CryptoUtils Module	41
5.4	Konfigurationsdatei	42
5.5	MVC-Muster	43
5.6	Http-Status code. RFC2616[2]	44
5.7	Status-Code Filter	45
5.8	Caching configuration	45
5.9	Datenbank Schema	46
5.10	Cryptone Compilierung	48

Tabellenverzeichnis

2.1	Vergleich asymmetrische/symmetrische Verschlüsselung	6
5.1	Technologie-Überblick	37
5.2	Headers	38

1 **Einleitung**

Kapitel 1

1.1 Gliederung

Diese Arbeit lässt sich in drei große Abschnitte unterteilen: Kapitel 2 behandelt die Anforderungen eines sicheren Dokumentenaustauschs sowie der Authentifizierungsmechanismen, die für das Verständnis der weiteren Kapitel wichtig sind. Im folgenden Kapitel 3 wird beispielhaft der aktuelle Stand der Technik vorgestellt und ihre technische Umsetzung aufgeführt. In Kapitel 4 wird anhand der Problemstellung ein Konzept für die Dokumentaustausch-Plattform erstellt, welches in den Kapiteln 5 und 6 konkretisiert und implementiert wird. Die letzten beiden Kapitel 7 und 8 fassen die Ergebnisse dieser Arbeit zusammen und machen Vorschläge für eine Verbesserung des Systems.

1.2 Problemstellung und Motivation

Der Austausch von vertraulichen Informationen mittels schriftlicher Aufzeichnungen ist grundsätzlich problematisch. Wie können Informationen zwischen Parteien ausgetauscht werden, ohne dass Unberechtigte diese mitlesen können. Die Lösung des Problems besteht darin, die Nachricht verschlüsselt zu übertragen. Das heißt, die ursprüngliche Nachricht wird so verändert, dass es Unberechtigten deutlich erschwert wird, den Inhalt einer abgefangenen Nachricht zu erfassen. Bereits in der Antike wurden vertrauliche Informationen verschlüsselt übermittelt. Schon damals bestanden die folgenden Schwierigkeiten, die noch heute trotz aufwendigerer Verschlüsselung relevant sind :

1. Wer kann Nachrichten ver- bzw. entschlüsseln, und wie? (Authentifizierung)
2. Wie werden die Schlüssel zwischen Sender und Empfänger ausgetauscht? (Kanalproblematik)
3. Wie wird sicher gestellt, dass die Nachricht den Empfänger so erreicht, wie sie geschickt wurde? (Integritätsprüfung)

Die Notwendigkeit eines sicheren Dokumentenaustauschs hat sich in den letzten Jahren als immer dringender erwiesen.

Dokumentenaustausch gibt es schon längst innerhalb geschlossener Netzwerke (Intranet). Wobei manche Sicherheitsaspekte wie Datenverschlüsselung oder auch Dateninte-

gritat absichtlich weggelassen werden, da man davon ausgeht, dass alle Benutzer des Intranets sich innerhalb des Unternehmens befinden und folglich vertrauenswurdig sind.

Intranet wird aufgrund seiner Eigenschaft vor auerer Gefahr geschutzt. Problematisch wird es aber, wenn der Datenaustausch uber ein offenes, unsicheres Netzwerk (Internet) geschehen soll. Warum soll nun diese Notwendigkeit fur Unternehmen bestehen? Man denke etwa an einen externen Mitarbeiter, der sich nicht immer innerhalb des Unternehmens befindet und trotzdem seine Projektpartner uber den Stand seiner Arbeit auf dem Laufenden halten mochte. Oder auch an ein Projekt, das von zwei oder mehreren Unternehmen durchgefuhrt werden muss. Dabei konnen die Unternehmen keinen gegenseitigen Zugriff auf ihr jeweiliges Intranet gewahrleisten. Die geschickte Losung fur diesen Fall ware Internet; aber spatestens durch die NSA-Affare ist es deutlich geworden, dass eine Datenubertragung von vertraulichen Informationen via Internet ohne weitere Sicherheitsmanahmen nicht geeignet ist.

1.3 Zielsetzung

Anhand des Abschnitts 1.2 stellt man fest, dass in Unternehmen die verbreitete Losung fur den Dokumentenaustausch per Intranet erfolgt. Diese Losung schutzt gegen auere Gefahren, weil Intranet ein geschlossenes Netzwerk ist. Heutzutage ist aber ein gemeinsamer Zugriff auf digitale Informationen nicht nur innerhalb, sondern auch uber Unternehmensgrenzen hinweg mit Partnern oder externen Mitarbeitern ein wichtiges Instrument geworden. Eine naive Losung durch Internet ohne weitere Sicherheitsmanahme ware fur Unternehmen gefahrlich.

Das Ziel dieser Arbeit ist, ein Internet-basiertes sicheres Datenaustausch-System zu entwickeln, das erstens die Komplexitat (Schlusselverwaltung, Schlusselubermittlung) vollstandig auf die Software delegiert. Zweitens muss gewahrleistet sein, dass keine zusatzliche Softwareinstallation notig wird, da die Software uber einen Webbrowser lauft. Im Unterschied zu etablierten Losungen wie Dropbox werden alle Daten vorab lokal verschlusselt, bevor das Hochladen geschieht.

2 Kapitel 2 Grundlagen und Stand der Technik

2.1 Begriffe

Schlüsselaustausch

Der Schlüsselaustausch ist von großer Bedeutung, was die Netz- und Informationssicherheit angeht. Auch bei etablierter Sicherheits-Software ist Schlüsselaustausch problematisch. Aufgrund ihrer Sensibilität gehören Chiffrierschlüssel zu kritischen Informationen.

Kritische (sensible) Informationen

Es handelt sich hierbei um Informationen bzw. Daten, die auf keinen Fall irgendwo in den verschiedenen Softwarekomponenten unverschlüsselt abgespeichert oder unverschlüsselt durch das Netz geschickt werden dürfen. Zu dieser Kategorie gehören beispielsweise wichtige Benutzerdokumente oder Benutzeranmeldeinformationen. Solche Informationen werden immer signiert, bevor sie gespeichert werden.

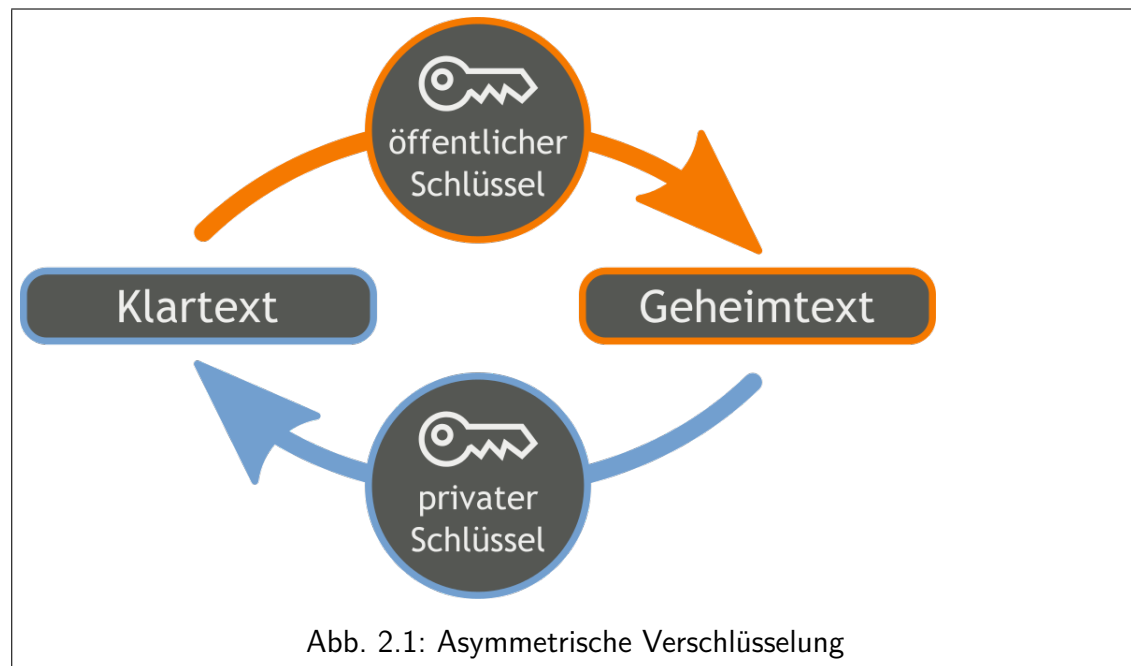
Schlüssel

Hierunter versteht man ein kryptographisches Werkzeug oder anders ausgedrückt ein Mittel zum Chiffrieren bzw. Dechiffrieren. Dieses kann verschiedene Formen haben und je nach Schlüsselart entweder zu den kritischen oder nicht kritischen Informationen gehören. Bei der Behandlung der einzelnen Schlüsselarten wird jeweils kenntlich gemacht, ob sie als kritisch oder nicht kritisch einzuschätzen sind.

Asymmetrische Kryptographie

Asymmetrische Kryptographie ist ein kryptographisches Verfahren, bei dem die kommunizierenden Parteien keinen gemeinsamen geheimen Schlüssel zu kennen brauchen.

Ein Benutzer erzeugt hier ein Schlüsselpaar 2.1, das aus einem geheimen Teil (privater Schlüssel) und einem nicht geheimen Teil (öffentlicher Schlüssel) besteht. Der öffentliche Schlüssel ermöglicht jedem, Daten für den Inhaber des privaten Schlüssels zu chiffrieren, dessen digitale Signature 2.1 zu prüfen oder ihn zu authentifizieren. Der private Schlüssel aber ermöglicht es seinem Inhaber, mit dem öffentlichen Schlüssel chiffrierte Daten zu entschlüsseln, digitale Signaturen zu erzeugen oder sich zu authentifizieren. Für die Fertigung dieser Arbeit wird der RSA-Algorithmus als asymmetrisches Chiffrierverfahren eingesetzt.



Assymetrische Schlüssel (Schlüsselpaar)

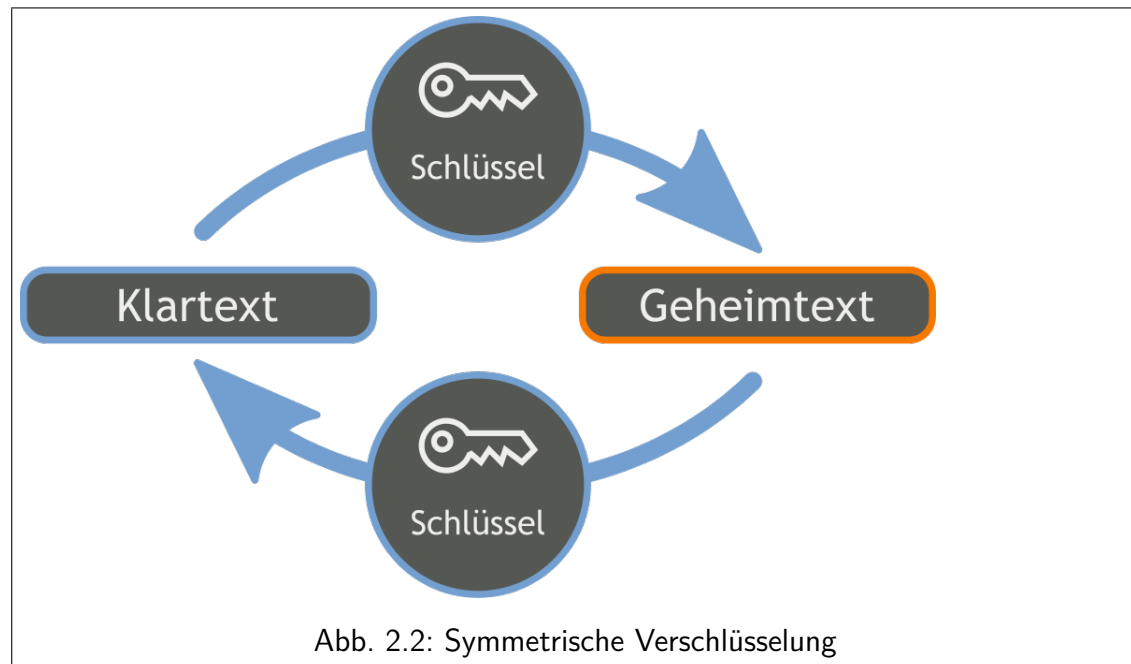
Beim RSA-Algorithmus werden Schlüsselpaare benötigt. Schlüsselpaare bestehen aus zwei Schlüsseln, einem geheimen und einem öffentlichen Schlüssel. Öffentliche Schlüssel werden eingesetzt, um Chiffrierung durchzuführen. Sie zählen nicht zu den kritischen Informationen. Mit geheimen Schlüsseln dagegen führt man die Dechiffrierung durch. Geheime Schlüssel, auch private Schlüssel genannt, gehören zu den kritischen Informationen.

Benutzer-Masterkey

Benutzer-Masterkey ist ein asymmetrischer Schlüssel der quasi eine „Generalschlüssel“-Funktion besitzt. Alle asymmetrischen Dokumentschlüssel werden mit dem Masterkey verschlüsselt bzw. entschlüsselt. Der geheime Teil des Masterkeys wird seinerseits noch einmal mit einer Passphrase verschlüsselt, welches nur der Benutzer kennt.

Symmetrische Kryptographie

Symmetrische Kryptographie ist ein Kryptosystem, beim welchem im Gegensatz zu einem asymmetrischen Kryptosystem beide Teilnehmer denselben Schlüssel (Symmetrische Schlüssel 2.1) verwenden. Bei manchen symmetrischen Verfahren (z.B. IDEA) sind die beiden Schlüssel nicht identisch, doch kann leicht der einer aus dem anderen abgeleitet werden. Für diese Arbeit wird exklusiv der AES-Algorithmus als symmetrisches kryptographisches Verfahren benutzt, wobei die Chiffrierschlüssel und Dechiffrierschlüssel gleich sind.



Symmetrische Schlüssel

Es handelt sich um einen geheimen Schlüssel, der beim AES-Algorithmus (Symmetrische Verschlüsselungsverfahren) eingesetzt wird, um Chiffrierung und Dechiffrierung durchzuführen. Da die symmetrischen Schlüssel sowohl zur Chiffrierung als auch zur Dechiffrierung eingesetzt werden, sind sie als kritische Informationen zu sehen.

Dateischlüssel

Ein Dateischlüssel ist ein symmetrischer Schlüssel, der eingesetzt wird, um Dateien zu chiffrieren bzw. zu dechiffrieren. In dieser Arbeit entspricht der Dateischlüssel den Gruppenschlüssel, kurz SGK (Secret Group Key).

Passwort und Passphrase

Duden [3] definiert das Passwort bzw. Passphrase wie folgt :

„nur Eingeweihten bekannte, aus Buchstaben, Ziffern oder Sonderzeichen bestehende Zeichenfolge, die den Gebrauch einer Sache, den Zugang zu ihr ermöglicht und sie gegen den Missbrauch durch Außenstehende schützen soll.“

Der Begriff „Passwort“ wird in der Dokumentation benutzt im Zusammenhang mit Benutzeranmeldeinformationen, und wird in einer modifizierten Form im RemoteServer gespeichert. Hingegen wird Passphrase außerhalb des LocalServers nie persistent gehalten. Das Passphrase wird angewandt, um den Benutzer-Masterkey zu verschlüsseln.

Digitale Signature

Eine digitale Signatur, auch digitales Signatur-Verfahren, ist ein asymmetrisches Kryptosystem, bei dem ein Sender mit Hilfe eines geheimen Singnatureschlüssels (dem privaten Schlüssel) zu einer digitalen Nachricht einen Wert berechnet, der ebenfalls digitale Signatur benannt wird. Dieser Wert ermöglicht es jedem, mit Hilfe des öffentlichen Verifikationsschlüssels (dem öffentlichen Schlüssel) die nicht bestreitbare Urheberschaft und Integrität der Nachricht zu prüfen. Um eine mit einem Singnatureschlüssel erstellte Signatur einer Person zuordnen zu können, muss der zugehörige Verifikationsschlüssel dieser Person zweifelsfrei zugeordnet sein.

Hybride Verschlüsselung

Hybride Verschlüsselung ist eine Kombination aus asymmetrischer Verschlüsselung und symmetrischer Verschlüsselung. Dabei wählt der Sender einen zufälligen symmetrischen Schlüssel, der Session-Key genannt wird. Mit diesem Session-Key werden die zu schützenden Daten symmetrisch verschlüsselt. Anschließend wird der Session-Key asymmetrisch mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Dieses Vorgehen löst das Schlüsselverteilungsproblem und behält dabei den Geschwindigkeitsvorteil der symmetrischen Verschlüsselung.

	asymmetrische Verschlüsselung	symmetrische Verschlüsselung
Effizienz	langsam	schnell
Schlüsselaustausch	elegant	problematisch

Tab. 2.1: Vergleich asymmetrische/symmetrische Verschlüsselugn

Wie die Abbildung oben zeigt, sind symmetrische Verschlüsselungsverfahren auch bei großen Datenmengen sehr schnell, asymmetrische dagegen sind sehr langsam, und deshalb nur für kleine Datenmengen (wie etwa für Schlüssel) geeignet.

Bei der Schlüsselverteilung hat das symmetrische Verschlüsselungsverfahren den Nachteil, dass sich die Kommunikationspartner vor der Übermittlung der Nachricht auf einen geheimen Schlüssel einigen müssen. Dazu muss ein sicherer Kommunikationskanal benutzt werden, wie zum Beispiel ein Kurier. Asymmetrische Verschlüsselungsverfahren dagegen lösen das Problem sehr elegant, weil zum Verschlüsseln nur der öffentliche Schlüssel gebraucht wird. Zur Übermittlung dieses Schlüssels reicht ein authentifizierter Kanal aus.

Hybride Verschlüsselungsverfahren kombinieren die beiden Verfahren (asymmetrisch und symmetrisch) so, dass deren Vorteile erhalten bleiben:

- Hybride Verschlüsselungsverfahren sind sehr schnell und eignen sich für große Datenmengen, weil die Daten mit dem symmetrischen Verfahren verschlüsselt werden und das asymmetrische Verfahren nur für den Sitzungsschlüssel verwendet wird.
- Es muss vor dem Senden der Nachricht kein geheimer Schlüssel ausgetauscht werden. Die Kenntnis des öffentlichen Schlüssels des Empfängers reicht, um zu verschlüsseln.

Public Key Infrastructur (PKI)

Public Key Infrastructur ist ein System, das digitale Zertifikate ausstellen, verteilen und prüfen kann. Die innerhalb einer PKI ausgestellten Zertifikate werden zur Absicherung Rechner-gestützter Kommunikation verwendet.

Basis für Public Key Infrastructur sind das asymmetrische Kryptosystem und die hybride Verschlüsselung. Mit Hilfe des asymmetrischen Kryptosystems werden die Daten digital signiert und verschlüsselt.

2.2 Email

Der Austausch von elektronischen Daten per Email ist weit verbreitet. Email-Dienste werden von eigenen Unternehmen oder auch von Internet-Dienstleistern angeboten. Die Versendung von Emails über das Internet ist zunächst unverschlüsselt. Somit sind die Inhalte relativ einfach auch für Dritte lesbar.

Die Identitäten der Kommunikationspartner werden i. Allg. mit den Email-Adressen gleichgesetzt. Eine weitergehende Prüfung findet nicht statt. Dadurch aber wird der Identitätsdiebstahl erleichtert, das heißt die Vortäuschung einer fremden Identität (Email-Adresse). Obwohl in der meisten Email-Clients eine Funktion vorgesehen ist, um das Risiko des Identitätstäuschung zu umgehen, geschieht die Email-Adressenüberprüfung nicht automatisch. Dies bleibt dem Benutzer überlassen.

Ein Austausch über Firmengrenzen hinweg ist problemlos möglich. Problematisch ist aber der Austausch von kritischen Informationen.

Eine auf dem ersten Blick triviale Lösung, was das Austauschen von sensiblen elektronischen Daten angeht, besteht darin, die Daten zu verschlüsseln und diese per Email an den Kommunikationspartner zu senden. Diese Lösung ist nur möglich, sofern sich die Teilnehmer (Sender und Empfänger) mit Kryptographie bzw. Kryptographiesoftware auskennen.

Überhaupt stellen sich dieser Lösung etliche Hürden in den Weg:

- Wie gerade erwähnt, setzt diese Lösung voraus, dass sich die Kommunikationspartner mit der Kryptographie bzw. Kryptographiesoftware auskennen.
- Zusätzliche Softwareinstallationen sind zwingend erforderlich.
- Der Schlüsselaustausch ist problematisch, und zwar insofern, als der Dechiffrierschlüssel an den Kommunikationspartner übermittelt werden muss.
- Dazu kommt noch die Infrastrukturproblematik: Schlüsselmanagement, Sicherheit der Schlüssel, Software-Installation etc. ...

2.3 Web-upload

Das Speichern von Dokumenten auf einem Internet-Server ist weit verbreitet und weltweit von jedem Browser aus möglich. Eine Installation zusätzlicher Software, oder gar die Öffnung zusätzlicher Ports der Unternehmens-Firewall ist nicht erforderlich. Die Benutzer-Authentifizierung erfolgt i.d.R. per Login/Password. Daten können im Internet mittels des http-Protokolls verschlüsselt übertragen werden. Fälschlicherweise wird angenommen, dass die übertragenen Dokumente dann auch beim Empfänger „sicher“ gespeichert sind. Jedoch werden lediglich die Dokumente auf dem Weg zum Server mit SSL verschlüsselt. Danach liegen sie zunächst unverschlüsselt vor. So werden von einem Server verschlüsselt übertragene Dokumente vom Browser entschlüsselt und im Klartext auf dem lokalen PC gespeichert. Ebenso werden Dokumente, die vom Browser für die Übertragung verschlüsselt werden, vom Server entschlüsselt und liegen dann am Server unverschlüsselt vor. Somit ergeben sich dieselbe Problematik und derselbe Lösungsansatz wie bei Datei-Servern. In Folge dessen sollten Dokumente, die per Browser auf einen Datei-Server geladen werden, vom Client-PC verschlüsselt werden. Die Dokumente müssen also vor dem Upload verschlüsselt worden sein, oder aber der Browser führt die Verschlüsselung durch. Eine Vorab-Verschlüsselung der Dateien hat den Nachteil, dass das Dokumenten- und Schlüssel-Management vom Anwender eigenverantwortlich durchgeführt werden muss. Dies ist den Anwendern i. Allg. zu aufwendig. Folglich sollte die Verschlüsselung durch den Browser quasi automatisch erfolgen. Dies wird

aktuell nur sehr selten durchgeführt, da die Verschlüsselungs-Softwares auch vom Web-Server geladen werden müssen. Und es kann nicht garantiert werden, dass die geladene Software nicht Eindringlingen unbeabsichtigten Zugriff ermöglicht. In Folge dessen werden Dokumente SSL-verschlüsselt zum Server gesendet. Die dort empfangenen, unverschlüsselten Dokumente werden sofort verschlüsselt und als Datei abgelegt.

Hier bestehen jedoch folgende Probleme:

- Wie kommen die notwendigen Schlüssel zum Server?
- Ein Eindringling auf dem Server kann die Klartext-Datei und/oder die Schlüssel mitlesen

Zusammenfassung: Ein Ansatz für ein sicheres web-upload ist bisher nicht bekannt.

2.4 File Transfer Protocol (FTP)

Das File Transfer Protocol (Dateiübertragungsprotokoll) ist ein in RFC 959 [4] spezifiziertes zustandsbehaftetes Netzwerkprotokoll zur Übertragung von Dateien. FTP läuft in der Anwendungsschichtmodell. Es wird benutzt um Dateien von Server zu Client herunterzuladen bzw. von Client zum Server hoch zu laden.

Eine wesentliche vorteilhafte Eigenschaft von FTP ist, dass die meisten Betriebssysteme (Unix basierende zum Beispiel) werden mit FTP-Client-und-Server vorinstalliert. Dadurch ist keine zusätzliche Softwareinstallation nötig.

Es existieren zahlreiche robuste FTP-Clients von Terminal-Client (mit Linux vorinstalliert) bis GUI-Client (Filezilla), welche die auch als Mozilla-Firefox oder Chrome Addons benutzt werden können.

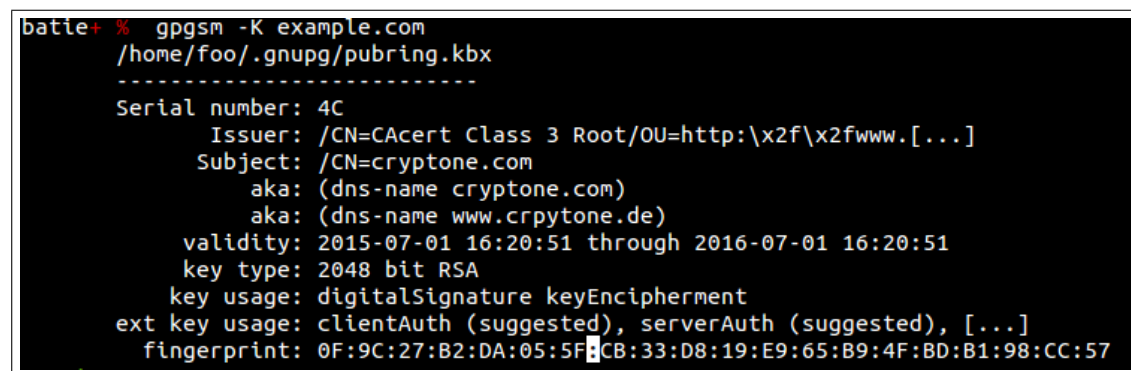
Mit dem Ziel, bessere Sicherheit zu gewährleisten, wurde SFTP (SSH Transfer Protocol) implementiert. SFTP ist eine alternative zum FTP. SFTP benutzt einen sicheren Kanal mit Hilfe von SSH für Dateitransfer.

2.5 Cloud-Service

Cloud-Service hat sich in den letzten fünf Jahren wesentlich verbreitet. Und war auf einem guten Weg bis zum NSA-Affäre sich als Standard einzusetzen. Heute auch trotz der Spionageskandale, wird Cloud-Service bei vielen Endbenutzern sehr beliebt. Gerade bei der Bereitstellung von sensiblen Dokumenten über die Cloud ist es unumgänglich, auf Sicherheit und Schutz vor fremdem Eingriff zu achten. Doch oft möchte man, einige Dateien mit einem Geschäftspartner oder Kollegen austauschen und achtet wenig auf Sicherheit. Beliebte Cloud-Lösungen aus dem Privatbereich wie Dropbox werden

hierbei gerne verwendet. Die Richtlinien solcher Cloud-Speicher entsprechen nicht dem deutschen Datenschutzrecht. Einer der eventuelle gravierende Problem ist es dass die Daten unverschlüsselt abgespeichert werden, Anders ausgedrückt, ein Dritter benötigt nur die Benutzer-Credentials, um das Dokument unverschlüsselt runter zu laden. Möchte man sein Dokument verschlüsselt hoch laden, muss man sich selber darum kümmern, dadurch entstehen die gleiche Problematiken wie beim Webupload2.3

2.6 PGP



```
batie+ % gpgsm -K example.com
/home/foo/.gnupg/pubring.kbx
-----
Serial number: 4C
    Issuer: /CN=CAcert Class 3 Root/OU=http:\x2f\x2fwww.[...]
    Subject: /CN=cryptone.com
           aka: (dns-name cryptone.com)
           aka: (dns-name www.cryptone.de)
    validity: 2015-07-01 16:20:51 through 2016-07-01 16:20:51
    key type: 2048 bit RSA
    key usage: digitalSignature keyEncipherment
    ext key usage: clientAuth (suggested), serverAuth (suggested), [...]
    fingerprint: 0F:9C:27:B2:DA:05:5F:CB:33:D8:19:E9:65:B9:4F:BD:B1:98:CC:57
```

Abb. 2.3: exemplarische zertifikat

PGP ist eine Daten-Ver- und Entschlüsselung Computerprogramm, das kryptographische Privatsphäre und Authentifizierung für die Datenkommunikation zur Verfügung stellt. PGP wird oft zum Signieren und zur Ver- und Entschlüsselung von Texten, E-Mails, Dateien und Verzeichnisse verwendet, um die Sicherheit der E-Mail-Kommunikation zu erhöhen. Es wurde von Phil Zimmermann im Jahr 1991 entwickelt.

Im PGP-System wird für einen Benutzer ein Schlüsselpaar (öffentlicher Schlüssel und privater Schlüssel) erzeugt. Dieses Schlüsselpaar ist mit einer eindeutigen ID verbunden, die normalerweise ein Name oder eine Email-Adresse ist. Die Schlüssel werden in einem Schlüsselbund Datensatz gespeichert. Der Eintrag eines öffentlichen Schlüssels im Schlüsselbund besteht aus einer ID, dem öffentlichen Schlüssel selbst und einem Zeitstempel, der auf das Erstellungsdatum des Schlüsselpaars referenziert. Öffentliche Schlüssel werden auf einem öffentlichen Schlüsselring gespeichert, wohingegen private Schlüssel auf einem privaten Schlüsselring gespeichert werden. Jeder Benutzer muss einen öffentlichen und privaten Schlüsselring speichern und verwalten[AR95][JF96]. Wenn Benutzer A eine gute Kopie des öffentlichen Schlüssels von Benutzer B besitzt, z.B. eine Kopie, deren er von der Integrität und Authentizität (keine Verfälschung etc.) überzeugt ist, dann kann A diese Kopie unterschreiben und an Benutzer C weitergeben. A wirkt somit als eine Mittelsperson von B zu C. Der von A signierte Schlüssel wird als Schlüssel Zertifikat bezeichnet. Jeder Benutzer muss im PGP-System erklären, welchen Personen er oder sie als Mittelsperson vertraut und muss den öffentlichen Schlüssel

der Mittelsperson mit seinem eigenen privaten Schlüssel signieren. Außerdem muss der Benutzer die verschiedenen Vertrauensgrade angeben, welche er zu seinen Mittelsperson hat. Eine Vertrauensbeziehung zu einer Person kann in Graden als unbekannt, nicht vertrauenswürdig, geringfügig vertrauenswürdig oder vollständig eingestuft, also klassifiziert werden. Jeder Benutzer speichert seine vertrauten Informationen oder Zertifikaten auf seinem in seinem PGP Konto. Abhängig vom Vertrauensgrad zu einer Mittelsperson ist dem entsprechenden Zertifikat im Schlüsselbund einen Gültigkeitsgrad zugewiesen. Er kann den Schlüssel in diesem Zertifikat nur dann verwenden, wenn der Gültigkeitsgrad hoch genug ist. Zum Beispiel kann ein skeptischer Anwender zwei vollständige Unterschriften für einen öffentlichen Schlüssel einfordern, um ihn als gültig anzusehen, wohingegen ein wenig skeptischer Benutzer, nur eine vollständig vertrauenswürdige Signatur oder zwei geringfügig vertrauenswürdige Signaturen verlangen könnte. Es ist wichtig zu beachten, dass Schlüsselringe und Vertrauensgrade es ermöglichen, jedem Benutzer seine eigene Vertrauenspolitik zu gestalten. Diese enge Vorstellung von Politik ist in PGP angebracht, denn es wurde speziell entworfen, um sichere E-Mails für den Einzelnen bereitzustellen. Die Unterschrift von A auf öffentlichen Schlüssel von B nicht so interpretiert werden sollte, dass A der persönliche Integrität von B vertraut. Die richtige Interpretation ist eher, dass A glaubt, dass die Bindung der Identität von B zum Schlüssel richtig ist. Darüber hinaus ist es wichtig zu beachten, dass das Vertrauen nicht transitiv ist. Die Tatsache, dass A dem B vollständig als Mittelsperson vertraut und dass B vollständig C vertraut, bedeutet nicht automatisch, dass A mit dem gleichen Grad C vertraut. Da PGP in der Popularität gewachsen ist, ist ein dezentrales "Web of Trust" entstanden. Jedes Individuum ist verantwortlich für den Erwerb der öffentlichen Schlüssel, die er braucht, und für die Zuordnung des Vertrauensgrads zu den Mittelpersonen, von denen er sie bekommt. Ähnlich muss jedes Individuum sein eigenes Schlüssel- paar erstellen und seinen öffentlichen Schlüssel verbreiten. Sein Ansatz lehnt folglich die Benutzung der offiziellen Zertifizierungsstellen ab, welche die öffentlichen Schlüssel eines Individuums unterschreiben. Damit handelt eine einzelne Person als "Vertrauensserver" für die Benutzer von diesen Schlüsseln. Ein Vorteil von PGP ist, dass jeder Benutzer denjenigen vertrauen (öffentlichen Schlüssel signieren) kann, denen er will. Außerdem bietet PGP die Möglichkeit, Gruppen zu erzeugen und in dieser Gruppe verschlüsselte Nachricht oder Dateien zwischen den Mitgliedern auszutauschen. Der erste Nachteil von PGP ist, dass die Software auf dem lokalen Rechner installiert werden muss und dort alle Schlüssel gespeichert werden. Wie soll der Benutzer dann seinen Schlüssel in einem anderen System oder in einer anderen IT-Infrastruktur benutzen. Die Schlüssel können zwar exportiert und importiert werden, jedoch führt dies zu einem erhöhten Aufwand. Ferner stellt sich die Frage, wie der Benutzer es einem Vertrauten ermöglicht, seine Schlüssel zu verwenden? Außerdem stößt die lokale Installation der Software auf bestimmte Anforderungen.

2.7 X.509

Der X.509[5] Authentifizierungsframework versucht, den gleichen Teil des Vertrauens-Management Problems wie PGP zu lösen, nämlich die Notwendigkeit, eine entsprechend zuverlässige vertrauenswürdige Kopie des öffentlichen Schlüssels einer Person zu finden, mit der man kommunizieren will. Wie in PGP, sind X.509-Zertifikate unterzeichnete Datensätze, welche die Benutzer ID mit ihrem kryptographischen Schlüssel assoziieren. X.509-Zertifikate enthalten weitere Informationen über PGP-Zertifikate, wie zum Beispiel den Namen des verwendeten Signatur-Verfahrens, um sie zu erstellen und das Zeitintervall, in dem sie gültig sind.

Aber ihr Hauptziel ist einfach die Bindung zwischen Benutzern zu ihren Schlüsseln zu schaffen. Jedoch unterscheidet sich X.509 scharf von PGP im Grad der Zentralisierung der Informationen. In PGP kann jeder öffentliche Schlüssel signieren und damit als Mittelsperson handeln. Der X.509 Framework fordert dagegen, dass jeder Benutzer seine Zertifikate von einer offiziellen Zertifizierungsstelle (CA) erhalten muss. Wenn Benutzer A ein Schlüsselpaar (öffentlicher Schlüssel, privater Schlüssel) erstellt, muss er es und die Rest der erforderlichen Informationen von einem oder mehreren CAs zertifizieren lassen und die erhaltenen Zertifikate in einem offiziellen Verzeichnisdienst registrieren. Wenn A später mit B sicher kommunizieren will, erhält er ein Zertifikat von B aus dem Verzeichnis-Server. Wenn A und B von der gleichen CA zertifiziert wurden, kann nur der Verzeichnisserver B's Zertifikat zu A senden. A kann dann die Gültigkeit dieser Zertifikat mit dem öffentlichen Schlüssel dieser gemeinsamen CA prüfen kann. Wenn A und B nicht unmittelbar durch eine gemeinsame CA zertifiziert werden, dann die Verzeichnisdienst müssen einen Zertifizierung-Pfad von A nach B erstellen. Um diesen Pfad zu verwenden, muss A den öffentlichen Schlüssel von der erste Zertifizierungsstelle in dem Pfad kennen. Somit beruht X.509 Framework auf der Annahme, dass CAs zu einem globalen Zertifizierungsstellen Baum-organisiert sind und dass alle Benutzer, die von CAs mit einem gemeinsamen Vorfahren in diesem globalen Baum unterzeichnet wurden[DWC03] [HPFS02] [CD03].

Das Problem ist, dass der Benutzer nicht eine autonome Identität einer weiteren Person prüfen kann, denn er ist vom öffentlichen Schlüssel seiner CA abhängig. Außerdem vertraut er automatisch alle Personen, denen die Öffentlichen Schlüssel durch die selbe Zertifizierungsstelle signiert wurden oder durch einer anderen vertrauten Zertifizierungsstelle. Dies stößt gegen einige Anforderungen, die fordern, dass die Benutzer nur gewünschte Personen vertrauen müssen. Darüber hinaus hat die NSA Affären bewiesen, dass Zertifizierungsstelle verfälschte Zertifikate erstellen können und somit werden verfälschte Identitäten freigegeben.

2.8 SPKI / SDSI

SDSI [EFL + 99][RL96] [ST00] wurde von Ronald Nieten und Butler Lampson konzipiert. Seine Entwicklung wurde durch die Komplexität der herkömmlichen Public-Key Infrastrukturen speziell die Abhängigkeit auf den globalen Namensraum motiviert. SDSI ist eine Public Key-Infrastruktur mit lokalen Namensraum. Dies macht es zu einem dezentralen Sicherheitssystem. SPKI wurde von Carl Ellison entwickelt und Andere. Es ist ein einfaches Autorisierungs-System. Hier wird der öffentliche Schlüssel nicht dazu genutzt, die Identität des Schlüsselbesitzers zu prüfen, sondern direkt seine Berechtigung zu bestimmten Diensten zu definieren. Die Vereinigung der beiden Projekte führt zum SPKI / SDSI, ein System zur Autorisierung und Authentifizierung. das SDSI lokalen Namensräume mit SPKI Autorisierung Systems kombiniert. SPKI / SDSI ist eine vollständig verteilte Lösung. Jeder Benutzer wird eine Zertifizierungsstelle und ist für die Verwaltung von Zertifikaten selbst zuständig. In SPKI / SDSI werden die Benutzer durch einen öffentlichen Schlüssel identifiziert und der öffentliche Schlüssel ist einem lokalen Benutzernamen Raum zugeordnet. Dieser Verein ist gültig lokal, das heißt, die zugehörigen Namen sind nicht global eindeutig. SPKI / SDSI erlaubt die Definition von Gruppen von Benutzern. Hier werden die öffentlichen Schlüssel der Mitglieder einer Gruppe des gleichen Namens zugeordnet. In SPKI / SDSI, gibt es zwei Arten von Zertifikaten: Namenszertifikate und Berechtigungszertifikate. Das Namenszertifikat bescheinigt, dass ein Name in einem Namensraum eines Emittent gültig ist. Das Berechtigungszertifikat gewährt den Ressourcenzugriff eines Benutzers. Namenszertifikate bestehen aus vier Bereichen zusammen: Issuer(Emittenten), Identifier(Identifizierter), Subject (Subjekt) und validity specification (Gültigkeit Spezifikation). Der Issuer ist derjenige, der das Zertifikat unterzeichnet. Der Identifizierter ist ein Byte, Zeichenkette, die einen Namen repräsentiert. Das Subject kann ein Name sein, oder ein öffentlicher Schlüssel. Wenn das Subject ein Name ist, ist es in lokalen Namenraum und der damit verbundene öffentliche Schlüssel kann wiederhergestellt werden. Schließlich Gültigkeit Spezifikation ist eine Gültigkeitsbedingung des Zertifikats, es könnte ein Gültigkeitsdatum oder eine Zugriffssteuerungsliste (ACL) sein. Berechtigungs Zertifikat besteht aus fünf Bereichen: Issuer(Emittenten), subject(Subjekt), Delegation, Tag, und ihrer Gültigkeit Spezifikation. Issuer und subject haben die gleiche Funktion wie oben geschrieben. Jedoch kann das Subject eine Gruppe von Nutzer sein. Das Feld Delegation zeigt an, dass das Zertifikat auch zu anderen Subjekten übertragen werden könnte. Tag gibt an, welche Berechtigungen empfangen wurden. Wie im Fall der Namenszertifikate ist Gültigkeit Spezifikation eine Gültigkeitsbedingung des Zertifikats [CEE + 01] [HM99]. Hier wird ein kurzes Beispielszenario genannt: Der Dateisystemressource Besitzer erzeugt zwei Berechtigungszertifikate. Ein Zertifikat ist mit der erteilten Zulassung: lesen, schreiben und nicht Delegation (RW: ND) zu D1 gegeben. Ein anderes Zertifikat ist an D2 mit Genehmigung : lesen und Delegation (R: D) gegeben. In der gleichen Figur

erzeugt D2 eine neue Zertifikat für D3 mit Leserechten, aber Delegation nicht erlaubt (R: ND). Wenn D3 auf das Dateisystem zugreifen muss, präsentiert er die Delegation Kette (D2 R: D - -> D3 R: ND) zu dem System.

Der Besitzer einer Datei will zum Beispiel den Zugriff auf die Dateien an viel Benutzer freigeben, dafür muss er jeweils ein Zertifikat mit den gewünschten Berechtigungen unterschreiben.

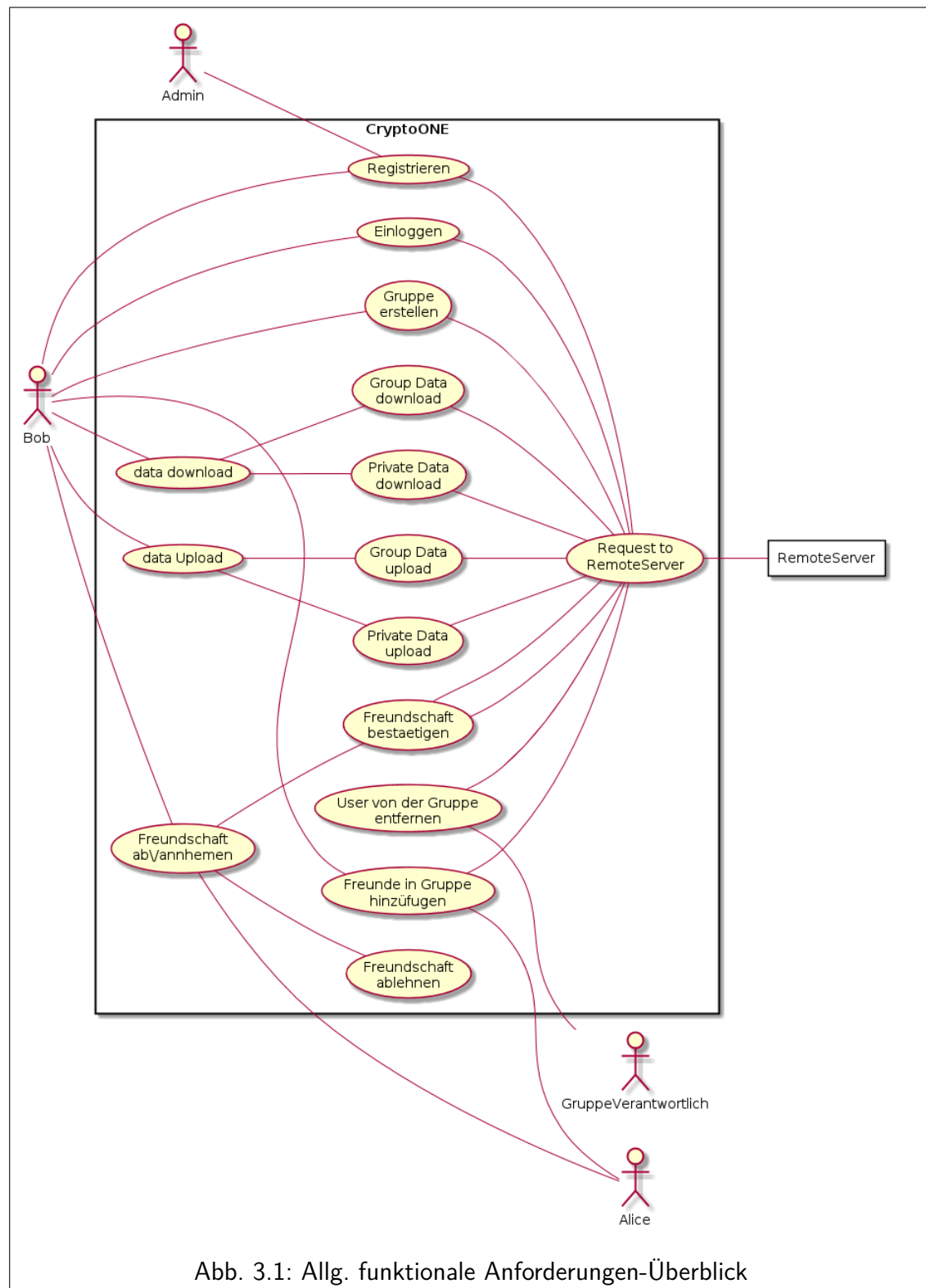
3

Kapitel 3

Anforderungen

Bei der Anforderungsanalyse unterscheidet man zwischen funktionalen und nicht funktionalen Anforderungen. Während funktionale Anforderungen den gewünschte Verhalten und die Funktionalität vorgeben, beschreiben nicht- funktionale Anforderungen Rahmenbedingungen wie Performance oder Zuverlässigkeit.

3.1 funktionale Anforderungen



Administrator Rolle

- Der Administrator muss in der Lage sein, neue Benutzer im System hinzu zu fügen und zu entfernen.

- Der Administrator darf nicht in der Lage sein, Benutzer kritische Informationen zu modifizieren oder zu lesen.

Benutzer role

- Der Benutzer kann eine Vertrauensbeziehung zu anderen Benutzern herstellen und sie wieder zurückziehen.
- Der Benutzer kann eine Gruppe bilden und wieder auflösen.
- Der Benutzer kann einer Gruppe vertraute Benutzer (Friends) hinzufügen.
- Der Benutzer kann den Zugriff auf seine Dateischlüssel an alle Mitglieder einer Gruppe freigeben und diese Freigabe auch wieder zurückziehen.

Registrierung und login

- Bei der Registrierung bzw. beim Login Phase dürfen kein Password oder keine Passphrase sowie Daten, die in irgendeiner Art mit dem Password bzw. mit der Passphrase korrespondieren, ins Netz gehen.

Server-Client Kommunikation

- Die Verbindung zwischen LocalServer und RemoteServer sollte Zustand-los sein.
- es darf keine SSL Verbindung zum Einsatz kommen.
- Server und Client müssen in der Lage sein sich gegenseitig zu authentifizieren.

3.2 Nicht-funktionale Anforderungen [1]

Nicht-funktionale Anforderungen sind in ISO/IEC 9126 definiert. Im Gegensatz zu funktionalen Anforderungen, die beschreiben was ein System leisten soll (funktional). geben die nicht-funktionalen Anforderungen an, Wie gut ein System etwas leisten soll (qualitativ). Die nicht funktionale Anforderungen sind in Rahmen des ISO Standards 9126 definiert. Dabei sind nicht-funktionalen Anforderungen Typen (Qualitätsattribute) wie Performance, Funktionalität, Usability, Portabilität, Sicherheit zu identifizieren.

3.2.1 Allgemeine nichtfunktionale Anforderungen

- Kein Einsatz von HTTPS

- RemoteServer darf keine Verschlüsselung bzw. Entschlüsselung durchführen
- LocalServer soll von ein USB-Stick gestartet werden, und soll auch von dort aus im Hintergrund laufen.
- Benutzerinteraktion erfolgt durch ein Browser so, dass keine zusätzliche Software-Installation erforderlich ist.

3.2.2 Wartbarkeit und Erweiterbarkeit

Die resultierende Software dieser Arbeit, soll in Zukunft gewartet, erweitert und verändert werden. Neu Features sind schon festgelegt (sollen aber in der jetzige Version nicht implementiert werden)

3.2.3 Portierbarkeit und Plattform-Unabhängigkeit

Defakto ist der LocalServer portierbar, LocalServer läuft auf USB-Stick . Localserver soll auch plattformunabhängig sein. Was RemoteServer angeht soll auch plattformunabhängig sein. Alle Einstellungen des Remoteserver müssen sich durch externe Konfigurationsdateien durchführen lassen.

3.2.4 Daten-und Serverintegrität

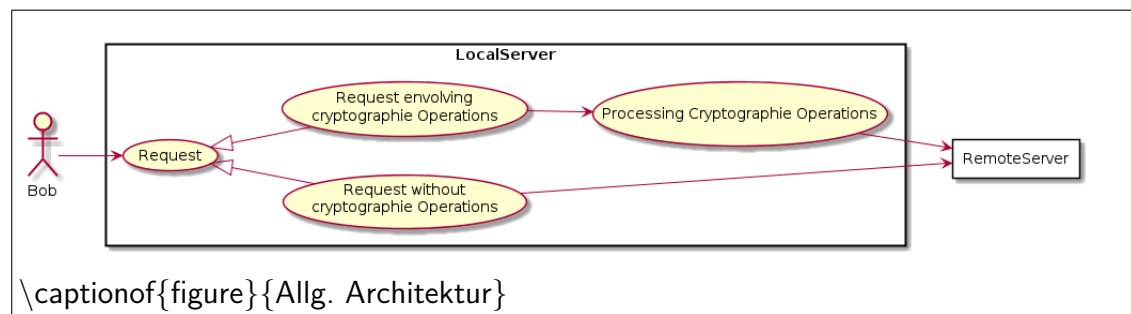
Der Benutzer soll in der Lage sein die Integrität von RemoteServer zu prüfen und der auf der letzter abgespeicherte Daten.

Kapitel 4

4 Konzept

4.1 Allgemein Architektur

Das System Architektur wurde als eine Server-Client Anwendung entworfen, LocalServer und RemoteServer, wobei der LocalServer eine Überbrückungsrolle zwischen den Frontend (Webbrowser Javascriptapplication) und RemoteServer spielt, LocalServer ist vergleichbar mit ein Proxy. Alle Chiffrierung-bzw-Dechiffrierungoperationen geschehen.



4.2 Authentifizierung

Authentifizierung spielt System-weit eine bedeutende Rolle. Dabei passieren alle notwendige Prüfung von [LocalServer], [RemoteServer], [RemoteServer] integrität und natürlich Verifizierung von Benutzer. Es durfte systemweit keine Einsatz von Zertifikat/SSL-Verbindung oder Aufbau eine zustandsbehaftete Verbindung kommen, spricht die Kommunikationskanal ist unsicher. Um Benutzercredentials von [LocalServer] auf [RemoteServer] zu übermitteln unter Anhaltung von Spezifikation, wurde SRP [6] (Secure Remote Password) eingesetzt.

Beim Einsatz von SRP-Secure Remote Password Protocol lässt sich auch einfach der gegenseitige Authentifizierung von [LocalServer] und [Remote-Server] realisieren, diese geschieht auch in der Authentifizierungsphase.

SRP Protokoll ist ein Authentifizierungsprotokoll, dabei können manche pro-Sitzung zufällige generierte Werte als Sitzungsschlüssel verwendet werden (B). Das bestehende Problem beim Einsatz von dauerhaften Passwörtern wird durch SRP minimiert, indem keine Passwort-korrespondierende (Hash, verschlüsselte Passwort) gespeichert wird, sondern ein „Verifier“. Wenn ein Angreifer die auf der Datenbank gespeicherten „Verifier“ , dann kann er nicht daraus die Benutzerpasswörter wiederrechnen. Ein gestohlener

Verifier ist ebenso nicht ausreichend zur Anmeldung, weil das Passwort immer noch benötigt wird.

Zur erfolgreichen Authentifizierung wird keine sensible Information ausgetauscht. „Sniffing-Attack“ ist dann dabei hilflos.

4.3 Kritische Daten Integrität

4.4 Schlüsselaustausch

Dokumentenschlüssel stehen nicht direkt in Verbindung mit Benutzer sondern mit der Gruppe. Wenn eine Gruppe kreiert wird, wird eine Gruppenschlüssel erzeugt. Dieser Schlüssel wird mit öffentliche Schlüssel der Gruppe-verantwortlich (GV) verschlüsselt. Jeder Benutzer der in der Gruppe eingeladen wird, bekommt dann eine Kopie der Gruppenschlüssel. Falls der zuvor eingeladenen Benutzer wieder durch der Gruppe-verantwortlich aus der Gruppe gelöscht wird, dann wird auch seiner Kopie der Gruppenschlüssel gelöscht sowie seine Verbindung zur Gruppe.

4.5 Server Integrität

Beim eine sicherheitsrelevante Anwendung ist es wichtig an jeder Anwendungsfälle an der Integrität der Softwareteile zu prüfen, sowie auch von exportierte Daten. Neben der Authentifizierung von Benutzer an sich, muss sich LocalServer an RemoteServer authentifizieren sowie RemoteServer an LocalServer.

RemoteServerintegritätsprüfung geschieht beim der Allererste Loginversuch, sowie beim jeder weiteren Request von LocalServer zu RemoteServer.

Dank SRP6-A Protokoll können LocalServer und RemoteServer sich gegenseitig authentifizieren, und zwar an Dritte Vorgang von der Protokoll.

Beim erfolgreichen Loginversuch wird ein Header `SERVER_PUBLIC_KEY` , die korrespondierte geheime Schlüssel muss nicht zwangsläufig geheim sein. An der LocalServer wird auch ein Schlüsselpaare erzeugt und der Header `CLIENT_PUBLIC_KEY` wird gesetzt. Diese weitere Massnahme verstärkt die Vertrauensbeziehung die beim erste gegenseitige Authentifizierung von LocalServer und RemoteServer etabliert wurde. LocalServer kann in weitere Request von LocalServer dann immer prüfen anhand von mit LocalServer private Schlüssel signierte Payload, die Payload authentifizieren (MAC), und dadurch auch sicherstellen dass der Request tatsächlich von LocalServer kommt.

4.6 Web-Of-Trust (Friends-Konzept)

Das Konzept des Web of Trust wurde von Phil Zimmerman für sein Programm „Pretty Good Privacy“ (PGP) entworfen, das mittlerweile zu OpenPGP [RFC 4880] weiterentwickelt wurde. Das Konzept basiert nicht auf der Existenz einer Vertrauenswürdigen Instanz, von der aus sich das Vertrauen automatisch transitiv ausgebreitet, sondern überlässt die Vertrauensbildung den Benutzern untereinander. Beim Einsatz dieses Konzept

wird kein Zertifikat wie in der Spezifikation [XXX] benötigt. Anstelle von Zertifikate werden ehe Öffentliche Schlüsseln benutzt. Ein in System System registrierte Benutzer bekommt eine Schlüsselpaare, wobei der private Schlüssel mit seinem Secret-Key verschlüsselt wird, und der öffentliche Schlüssel wird mit der private Schlüssel signiert. Zusammengefasst werden die folgende Informationen in der Datenbank gespeichert :

1. Benutzer Credentials
2. Public Key
3. Hashwert von Public Key
4. Public Key Signature
5. Private Schlüssel (verschlüsselt mit Secret-Key)
6. Hashwert von der privaten Schlüssel

Exemplarische Vertrauensbeziehung zwischen Zwei Benutzer:

```
1 {  
2   "id"      : 146,  
3   "created_at" : "2016-03-14T17:38:24.55",  
4   "friend_id" : 35,  
5   "signature" : "ITn...zGFzGsN/jMztHmLV2xjA+ZgjP5uslOYDP2WKC+HkZg7HfF==",  
6   "user_id"  : 24  
7 }  
8
```

Abb. 4.1: exemplarische Vertrauensbeziehung

Unterschied zu PGP bzw. OpenPGP wird eine weniger komplexer immerhin informative Daten abgepeichert zur Materialisierung von Vertrauensbeziehung zwischen zwei Benutzer.

Benutzer als Friend hinzufügen

Sei Alice und Bob zwei Benutzer des Cryptone System.

Alice möchte Bob als Freund hinzufügen. Eine Friend-Beziehung bzw. Vertrauensbeziehung zwischen Alice und Bob entsteht wenn das Public-Key von Bob von Alice signiert wurde und das Public-Key von Alice von Bob ebenso signiert wurde.

Die Benutzer die sich in System befinden können sich in eine Freundschaftbeziehung befinden (Friends).

Hat Alice Bob als „Friend“ hinzugefügt, so können weitere Freunde von Alice Bob als Friend annehmen. Dadurch bildet sich eine Kette von Friends [WEB-OF-TRUST]. Weitere ist die Freundschaftbeziehung zwischen Benutzer ein Randbedingung damit A beispielsweise B in eine Gruppe hinzufügen kann, und mit B dann auch Dokumenten

durch von A gegründete Gruppe austauschen kann.

Bei der Herstellung eine Vertrauensbeziehung (Friend) zwischen A und B , wird die öffentliche Schlüssel B von A signiert. diese Signatur bildet materialistisch die Vertrauensbeziehung zwischen A und B.

Die Signatur spielt nicht nur eine Rolle, um der Vertrauensbeziehungsherstellung zwischen zwei Benutzer, sondern auch beim Überprüfung von RemoteServer Integrität. Log sich ein Benutzer ein und merkt dass seine Signaturen nicht mehr stimmen, dann wurde der RemoteServer kompromittiert.

Vertrauensbeziehung zurückziehen

4.7 Übermittlung von Passphrase

Passphrase wird übers Handy durch RemoteServer an der LocalServer weitergeleitet. Dank diese Massnahme wird eine Gefahr abgelöst, und zwar das Gefahr dass ein Key-Logger auf der Computer installiert ist. Aus diesem Grund ist es sinnvoll das Passphrase durch eine andere Kanal zu übermittelt. Das ganze sollte keine zusätzliche Softwareinstallation benötigen, spricht die Übermittlung von Passphrase sollte per Browser geschehen, und zwar ein Handybrowser.

Wie in Abschnitt 4.5 gesehen, wird beim Start der LocalServer ein Schlüsselpaare erzeugt, der CLIENT_PUBLIC_KEY wird dann als Header gesetzt, und der geheime Schlüssel bleibt an LocalServer. diese öffentliche frisch generierte Schlüssel ist nicht mit der Benutzerschlüssel zu verwechselt. Die öffentliche Schlüssel der als Header gesetzt wurde und von daher systemweit zugreifbar ist kann an der Handy geschickt wird, und von dort aus wird von der Benutzer eingegeben Passphrase mit der CLIENT_PUBLIC_KEY verschlüsselt. die mit der öffentliche Schlüssel verschlüsselte Passphrase gehe dann von Handy zur LocalServer über RemoteServer.

4.8 Gruppe

Gruppe in Code Dokumentation „Group“ ist eine wichtige Konzept in Cryptoone-System. Gruppe ist das Konzept was Benutzer miteinander verbindet zugeordnetes Dokument und Schlüssel.

Zu eine Gruppe gehört eine Schlüsselkey (Symmetrische Schlüssel) , abgekürzt GK, da diese Schlüssel eine kritische Information ist, taucht der nicht unverschlüsselt in RemoteServer. Beim Erstellen einer Gruppe beim einem Benutzer wird der KG mit der öffentlichen Schlüssel der Benutzer verschlüsselt, zu einer Gruppe gehört genau einer KG.

Eine Gruppe kann genau einen (Private Gruppe PG) , genau zwei (Eins-zu-Eins-Gruppe EZEG) oder 1 bis mehrere Mitglieder (öffentliche Gruppe OG) haben.

Der Gruppeverantwortlich kann jederzeit der Gruppe löschen, und zwar ganz unabhängig von der Art der Gruppe.

4.9 Dokumentaustausch

von private Gruppe zu öffentliche Gruppe

das Teilen eines Dokuments lässt sich in verschiedene Weise durchführen. Ein Benutzer kann ein Dokument per Upload in seine private Gruppe hochladen. Das Dokument bleibt in seine private Gruppe solange bis der Benutzer die Entscheidung trifft das Dokument mit eine andere Gruppe zu teilen. Ab dann gehört das Dokument nicht mehr der Benutzer, dass heisst der kann das Dokument nicht mehr zurückziehen, ausgeschlossen der Benutzer ist der Gruppeverantwortlich der Gruppe, dann kann der wieder das Dokument zurückziehen. Das Original des von neulich in eine öffentliche Gruppe geteilte Dokuments bleibt aber im Benutzer private Gruppe.

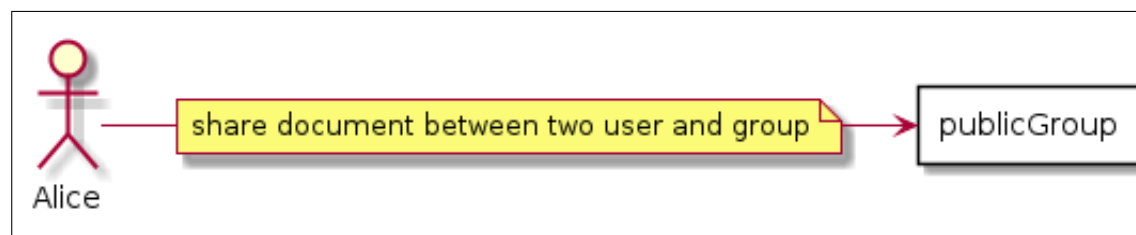


Abb. 4.2: Zwischen Benutzer und Group

zwischen zwei öffentliche Gruppe

Ein Dokument der sich in eine öffentliche Gruppe befinden kann mit einer anderen Gruppe geteilt werden, vorausgesetzt dass der Gruppeverantwortlich auch in der zweite Gruppe Mitglied ist.

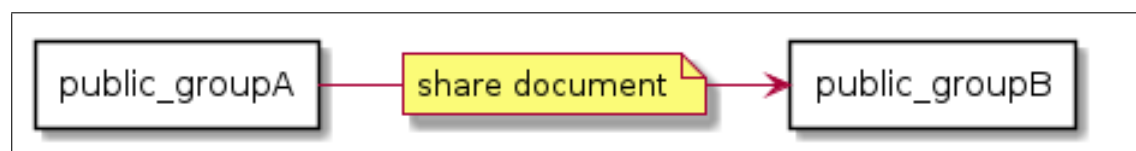


Abb. 4.3: Public Group zu Public Group

Ablauf :

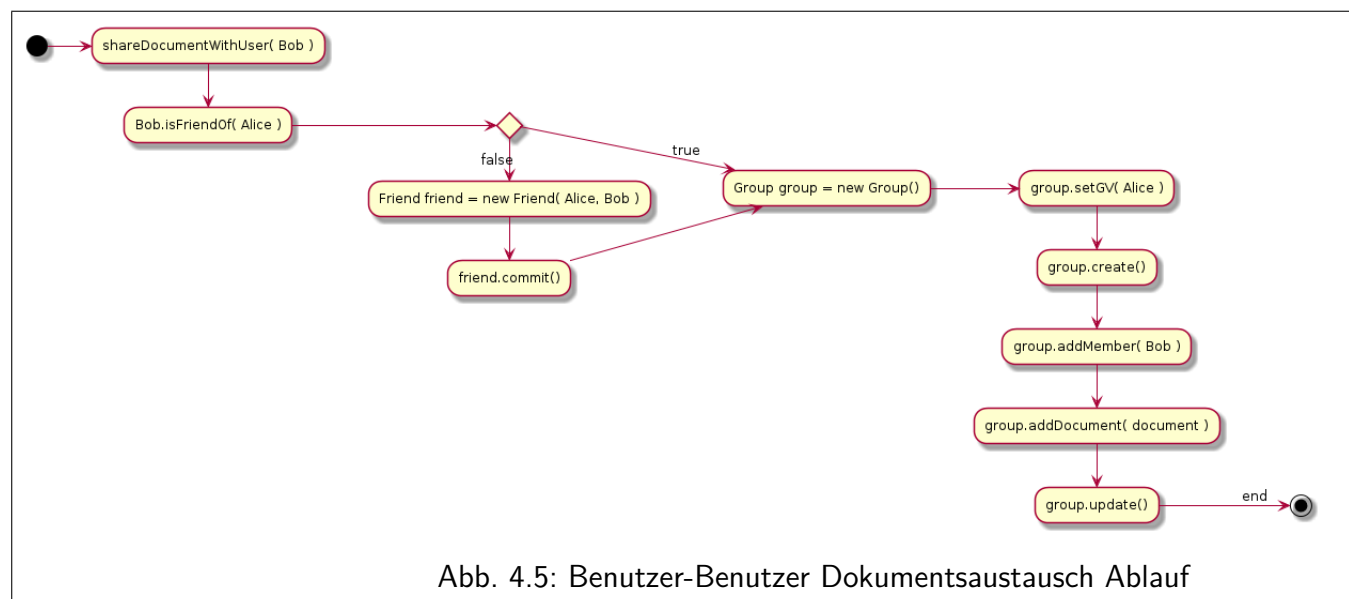
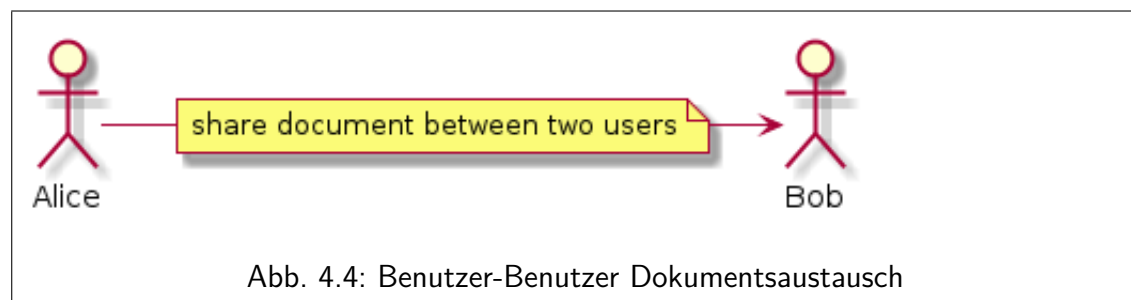
1. Dokument D in Gruppe A

2. Prüfen ob Benutzer U GV in Gruppe A
3. Prüfen ob GV , Gruppemitglieder in Gruppe B
4. authorisieren das Teil von Dokument D zwischen Gruppe A und B

Zwischen zwei Benutzer

Ein Benutzer A kann sich entscheiden ein Dokument D nur mit einer Ihrer Freund B zu teilen. Ist der Benutzer B noch keiner Freund von A, dann wird während dieser Vorgang B als Freund von A vertraut gemacht (Das heisst während dieser Vorgang der Benutzer B öffentliche Schlüssel wird von Benutzer A signiert). Eine Gruppe wird dann erzeugt und das zu auszuteilenden Dokuments wird in der neulich erzeugte Gruppe verwiesen. der Benutzer B muss dann die Freundschaft anfrage von A bestätigen schließlich darf er auf Dokument D zugreifen.

Wie auf der unterstehende Abbildung ?? zu sehen ist, der Benutzer der den Vorgang auslöst wird dann als Gruppeverantwortlich der zu erzeugende Gruppe markiert, in der Fall Benutzer Alice. Alice kann jeder Zeit die Gruppe wieder löschen oder Bob von der Gruppe entfernen.



4.10 REST (Representational State Transfer)

Es darf systemweit HTTPS nicht im Einsatz kommen und/oder Einsatz von zustand behaftete Verbindung (Session). Ein sehr geschickte Konzept um diese Anforderung zu erfüllen ist der REST-Konzept. Der REST-Konzept besagt dass eine Anfrage alle Informationen zur vollständige Bearbeitung der Anfrage beinhalten muss. die Anfrage benutzen explizite Http Methode (GET, POST , PUT, DELETE), und die auf der Server zur Verfügung gestellte Resource haben eine Dokument ähnliche Organisation. Beim Einsatz von REST wird kein Query String benutzt, was als Vorteil hat, eine mögliche Sicherheitlücke zu vermeiden. Eine weitere Vorteil ist die Überschaubarkeit der Software.

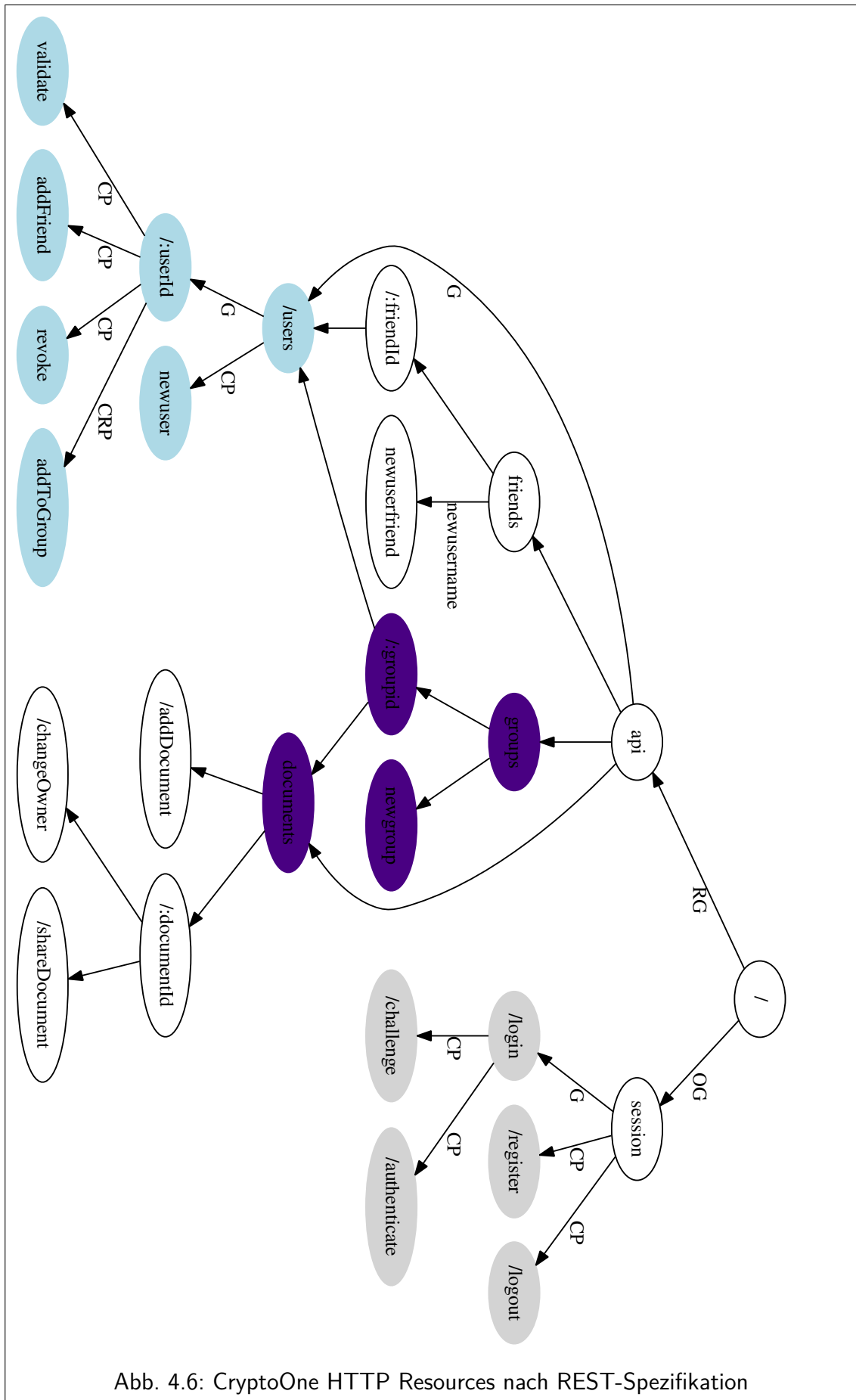


Abb. 4.6: CryptoOne HTTP Resources nach REST-Spezifikation

4.11 Integrität der transportierte kritischen Daten (JWS)

Bei sicherheitsrelevanten Software ist es wichtig die Daten zu verschlüsseln mit einem konsequent Schlüssellänge, und die Integrität diese Schlüssel gewährleistet. Diese gilt und ist ausreichend solange die Daten und Schlüssel nicht über Internet beispielsweise transportiert werden. Wenn die Daten über eine unsichere Netzwerk übermittelt werden muss eine zusätzlicher Integritätsmechanismus im Spiel kommen, um sicherzustellen dass eine Daten die von A zu B beispielsweise transitiert, unterwegs nicht verfälscht wurde. Als Lösungssatz wird eine End-to-End Integritätsprüfung eingesetzt mithilfe der JWS (JSON Web Signature) spezifiziert in RFC 7515[7]

JWS basiert auf JSON Web Encryption (JWE)[8] um kryptographische Operationen durchzuführen. JWE implementiert die meistens bekannte standard kryptographie Algorithmus, wie RSA, AES

JWS benutzt die von JWE implementiert RSA um Daten zu signiert. Bei Einsatz von JWS muss Acht darauf gegeben dass die JSON-Daten richtig formatiert wurde.

exemplarische JWT Headers:

	Headername	Beschreibung
1	alg	Name der kryptographische Algorithmus
2	hash	hash Wert
3	typ	Mediatyp
4	jwk	JSON Web Key

Abb. 4.7: JWS Header

JWS stellen weitere Headers zur Verfügung, aber die werden Anhand diese Arbeit nicht benutzt. Die Header werden gesetzt bei der letzte Filter. In diese Signature-Filter, wird die von Controller bereitgestellte Inhalt dessen Hash berechnet und gegebenenfalls auch signiert (nicht alle Daten müssen signiert werden).

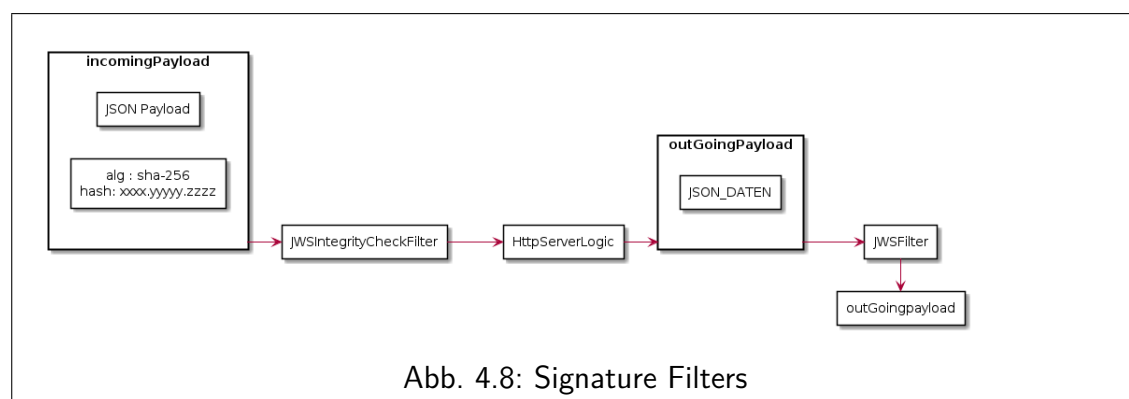


Abb. 4.8: Signature Filters

4.12 JSON Web Token (JWT)

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoiYXZgeFONFh7HgQ
```

Decoded

HEADER:

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD:

```
{  "sub": "1234567890",  "name": "John Doe",  "admin": true}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
) ☐ secret base64 encoded
```

Abb. 4.9: JWT-Token : Erläuterung

JWT ist eine JSON-basierte Standard um Authentication-Objekt zu repräsentieren. Eine Anwendungsfall ist beispielsweise OpenID-Authentication. Es erlaubt einem Benutzer, der sich bei seinem sog. AuthenticationProvider einmal mit der Benutzername und Passwort angemeldet hat, sich nur mit Hilfe erhaltene JSON Web Token (JWT) ohne Benutzername und Passwort bei allen das System unterstützenden Websites anzumelden. wobei man sich gegen ein Authenticationserver A authentifizieren kann, den erhaltene Token wird dann später anhand eine Anfrage an ResourceServer benutzt.

Token werden an Stelle von Benutzername-Passwort-Kombinationen verwendet, um auf Ressourcen zuzugreifen. Ein Token ist meist eine Zeichenkette aus Buchstaben und Zahlen; Sonderzeichen können auch verwendet werden. Um es vor Missbrauch zu schützen soll es schwer zu erraten und passend zu einer Sicherheitsabfrage sein. OAuth unterscheidet zwischen Abfrage-Token und Zugangs-Token.

4.13 Notar (SGK zurückgewinnen)

In Falls das System wird unsicher, nach Entdeckung zum Beispiel eines verfälschten öffentlichen Schlüssel, wird alle Informationen in Datenbanken und dabei alle Schlüssel

gelöscht. Es sollte eine Möglichkeit gegeben, die Dateischlüssel zurückzugewinnen. Deswegen wird in System den öffentlichen Schlüssel eines Notars eingeführt. Bei Hochladen eines Dateien wird eine Kopie des symmetrischen Schlüssels des Dateien mit dem Notar öffentlichen Schlüssel verschlüsselt. Nach Meldung eines Problems im System, wie fehlerhaftes oder falsches Signatur beziehungsweise öffentlichen Schlüssel, können alle SGK anhand des Notar privaten Schlüssel entschlüsselt werden und damit alle Dateien zurückgewinnen. Hier ist es wichtig zu notieren, dass der Notar private Schlüssel ist nicht in System gespeichert, sondern in eine sichere Ort außerhalb des Systems. Nach Generierung ihres Schlüssel-paar müssen alle neuen Benutzer den Notar öffentlichen Schlüssel signieren und bei jeder Anmeldung seiner Integrität prüfen. Dadurch wird seine Integrität gewährleisten.

4.14 Szenarien

Hier werden einige Szenarien dargestellt und dazu sequentielle Diagramms, um die Interaktionen zwischen die einzelne Komponente zu verdeutlichen.

Benutzer registrieren

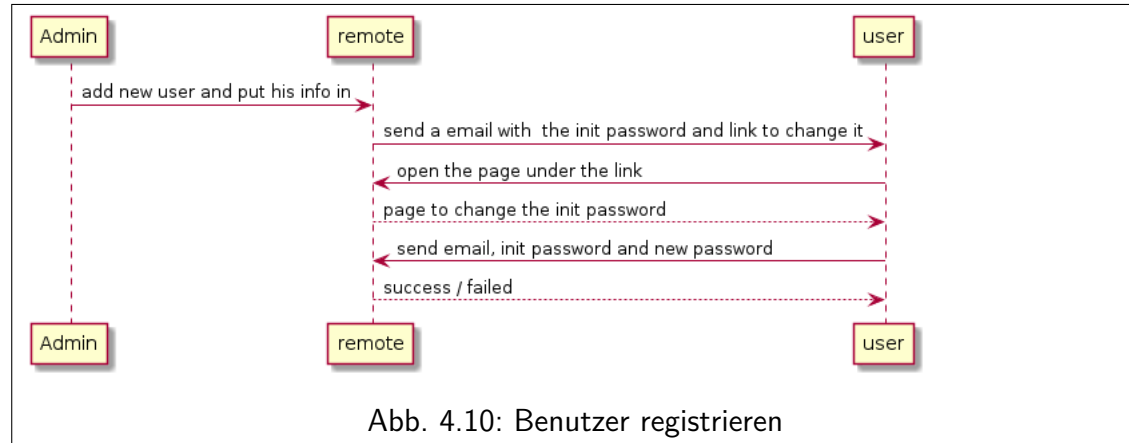
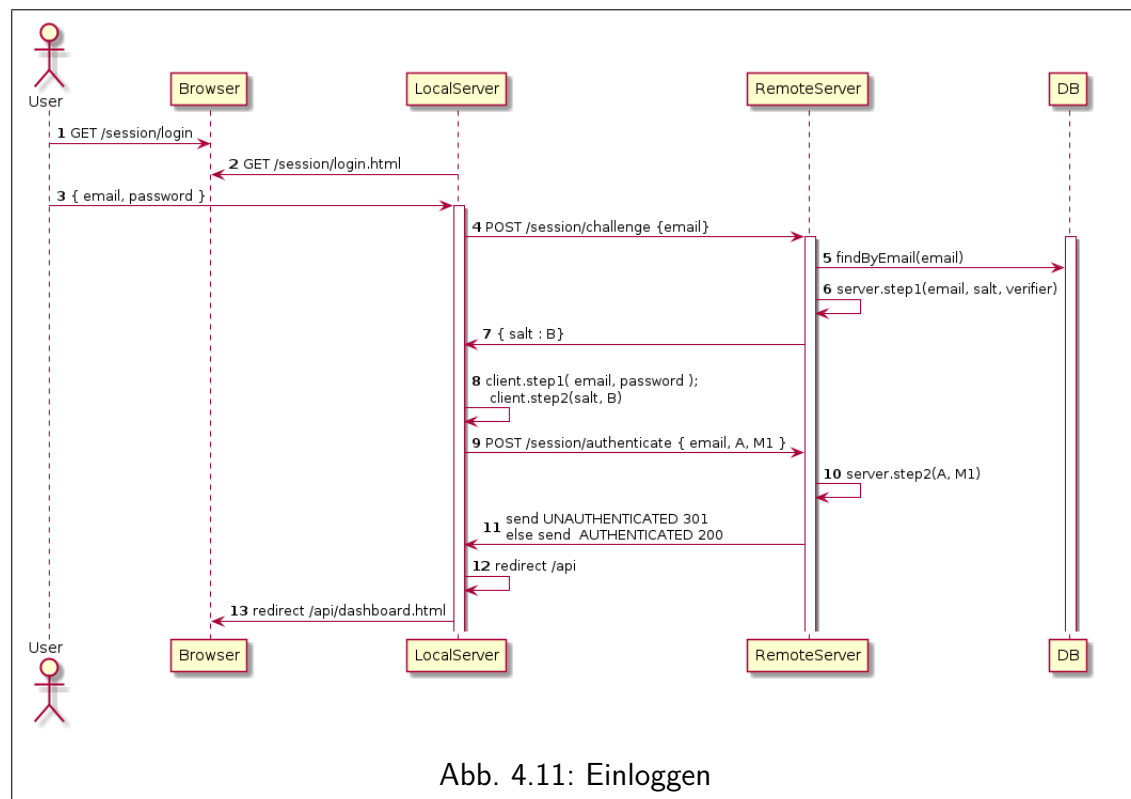
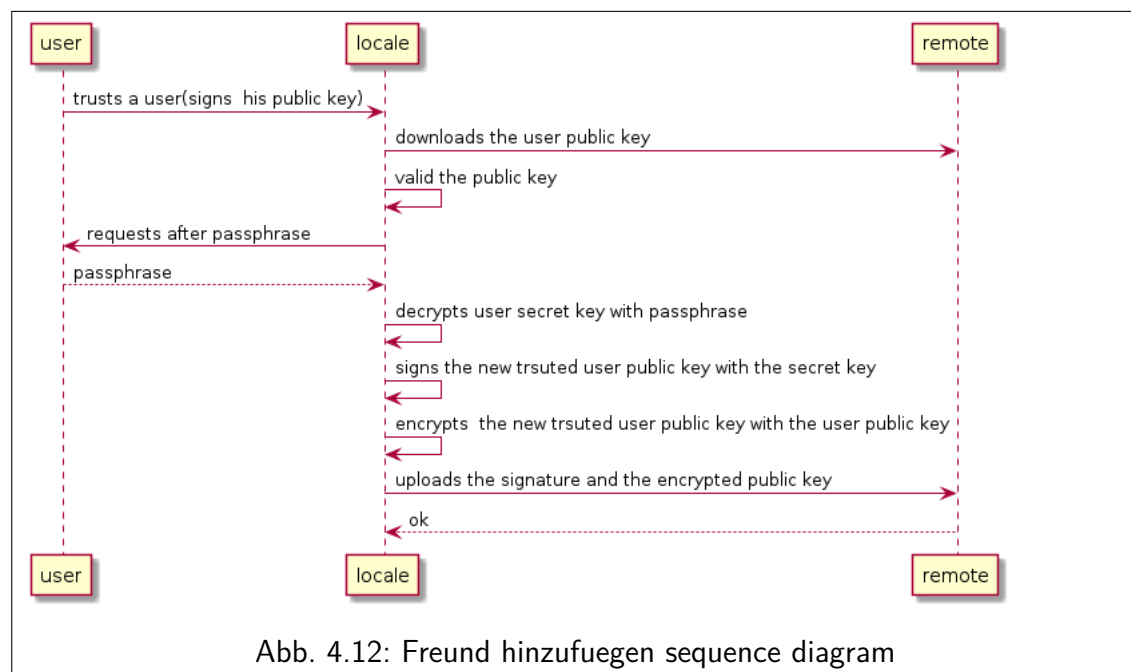


Abb. 4.10: Benutzer registrieren

Benutzer einloggen



Freund hinzufuegen

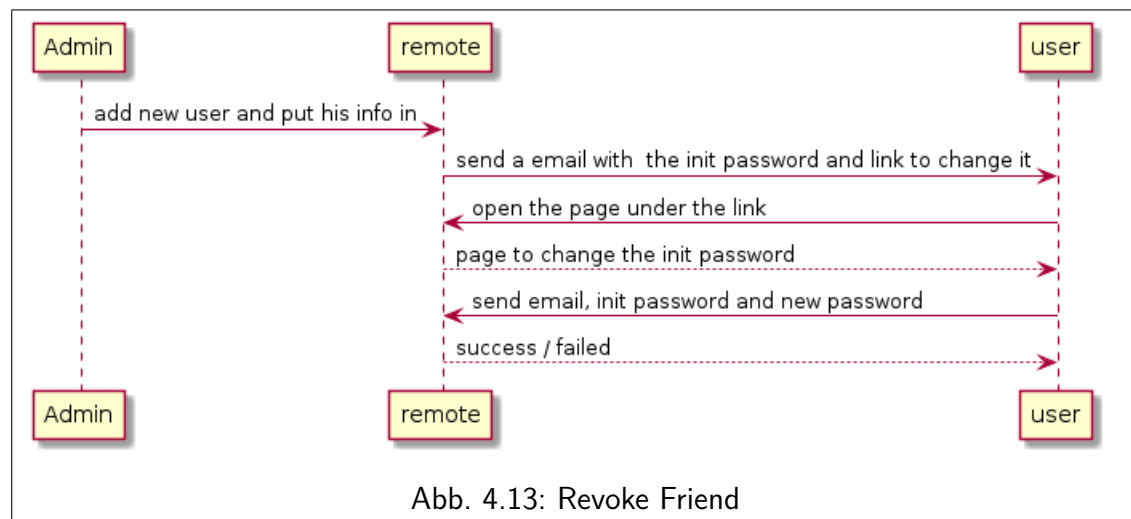


- Beispielszenario: Alice will Bob als Freund markieren
Alice fordert die Benutzerlistenseite auf, klicke auf den „+“ (Plus)-zeichen, um Bob als Freund hinzuzufügen.

- Alice ist gefordert, Ihre Passphrase einzugeben (Falls Passphrase noch nicht vorhanden ist)
- Der LocalServer lädt Bobs öffentlichen Schlüssel von RemoteServer herunter.
- LocalServer lädt Alice private Schlüssel vom RemoteServer herunter (Falls noch nicht vorhanden) und entschlüsselt der letzte mit Alice Passphrase
- Bobs öffentlichen Schlüssel wird mit Alices privaten Schlüssel signiert.
- Aus Signature, Alices und Bobs Id wird ein „Friend“ Objekt erzeugt. Dies entspricht die Vertrauensbeziehung zwischen Alice und Bob. Diese Objekt wird an RemoteServer geschickt und dort gespeichert.

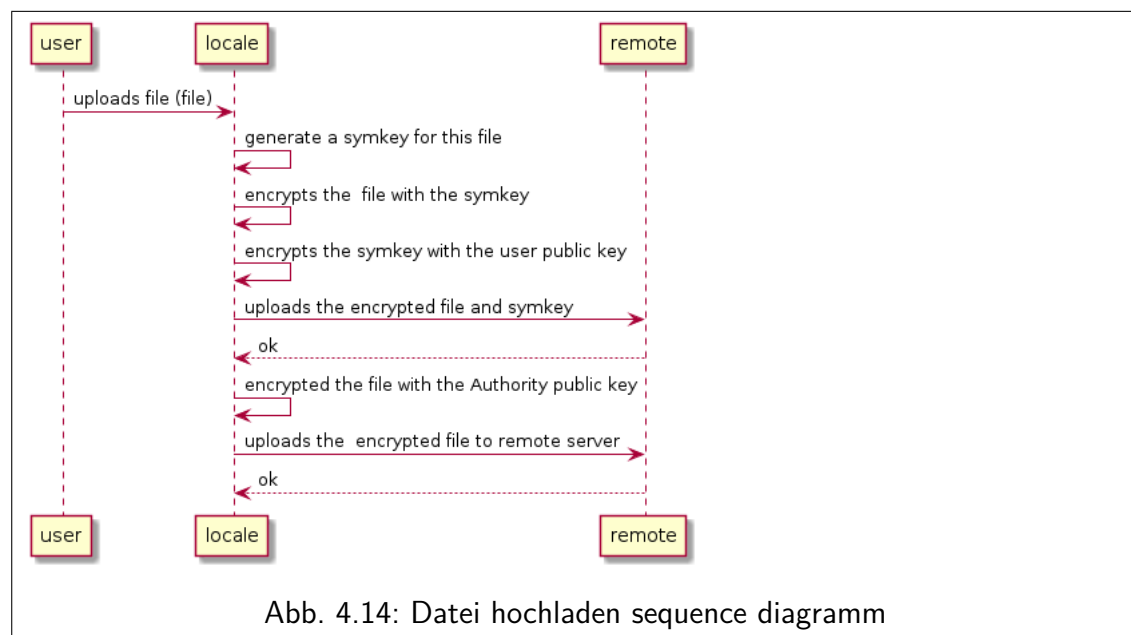
Freund revoke

Alice will Bob von seine Vertrauensbeziehungen rausnehmen



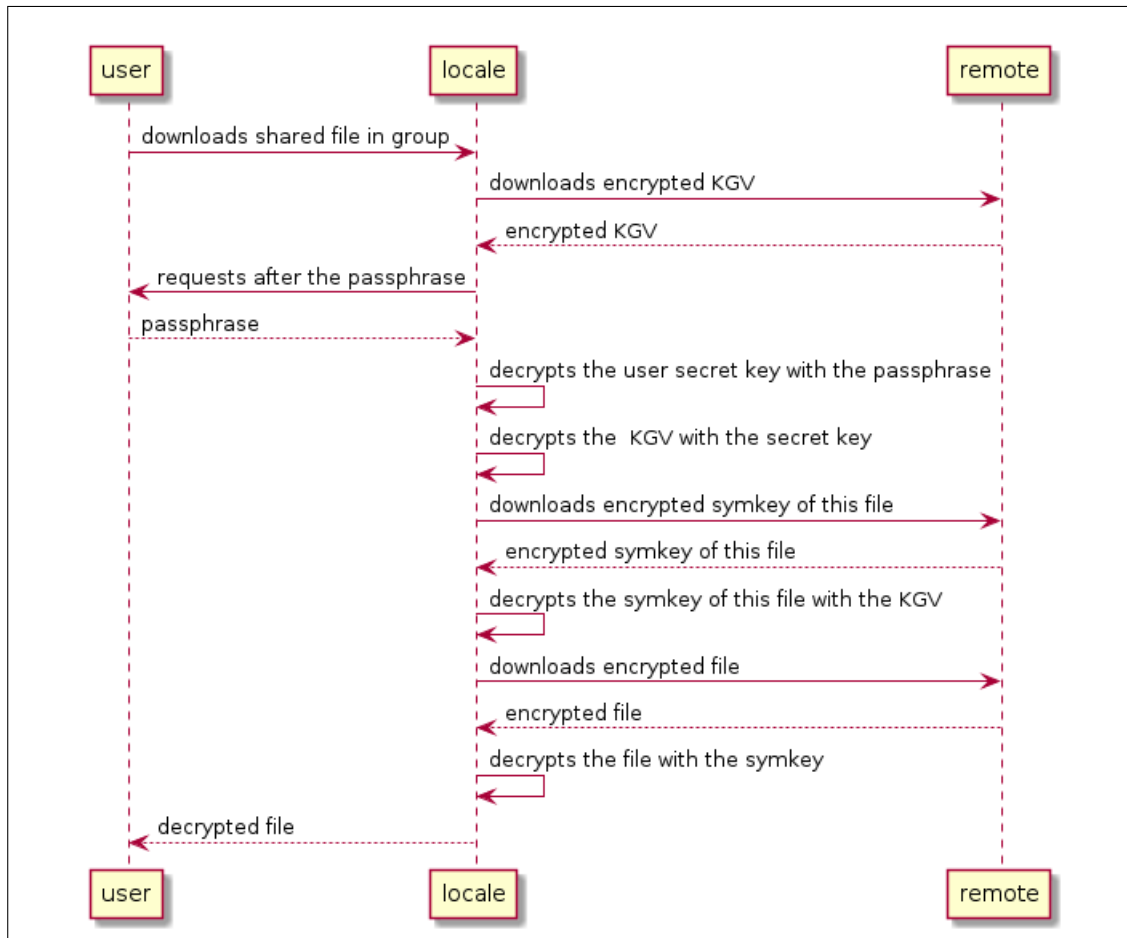
- Alice fordert die Freundelisteseite auf, klicke auf den „-“ (Minus)-zeichen, um Bob als Freund hinzuzufügen.
- Alice ist dann gefordert, die Aktion zu bestätigen
- Bestätigt Alice die Aktion, dann wird die revoke-Request über LocalServer zu RemoteServer weitergeleitet
- Auf der RemoteServer wird die Eintrag was die Freundschaft zwischen Alice und Bob gelöscht
- Alice wird über die erfolgreiche Zerstörung einer Vertrauensbeziehung mit Bob informiert.

Datei hochladen



- Alice will eine Datei hochladen
- Alice wählt sich eine Gruppe aus wo sie mitglied ist
- Alice clickt auf Datei Upload und wählt sich Datei aus, die sie gerne in die vorherige ausgewählte Gruppe hinzufügen will
- Alice Clickt auf dem Button „upload“ (hochladen)
- Die Anfrage wird an LocalServer weitergeleitet
- Alice Passphrase wird gefördert falls noch nicht im LocalServer vorhanden ist
- Alices Schlüsselpaare wird von RemoteServer runtergeladen sowie von Alice ausgewählte Gruppe Secret Key (SGK)
- Alice Kopie von SGK wird mit Alice private Schlüssel dechiffriert
- mit der dechiffriert SGK wird die Datei verschlüsselt. Dechiffriert SGK wird bald als die Verschlüsselung von Datei erfolgt hat von LocalServer zerstört.
- Schliesslich wird die chiffriert Datei zu RemoteServer übermittelt und ein Eintrag von Typ Dokuments auf der Datenbank gespeichert.

Datei herunterladen



- Alice befindet sich in einer Gruppe und will ein Datei runterladen. Also Alice muss mitglied von der Gruppe sein
- Alice click auf „download“ (herunterladen), Die Anfrage wird dann an LocalServer weitergeleitet
- Alice Passphrase wird gefördert falls noch nicht im LocalServer vorhanden ist
- Alices Schlüsselpaare wird von RemoteServer runtergeladen sowie von Alice ausgewählte Gruppe Secret Key (SGK)
- Alice Kopie von SGK wird mit Alice private Schlüssel dechiffriert
- LocalServer lädt von RemoteServer der geförderte Datei runter.
- Datei wird mit dechiffrierte SGK entschlüsselt.

Neue Gruppe erzeugen

Alice will eine neue Gruppe anlegen

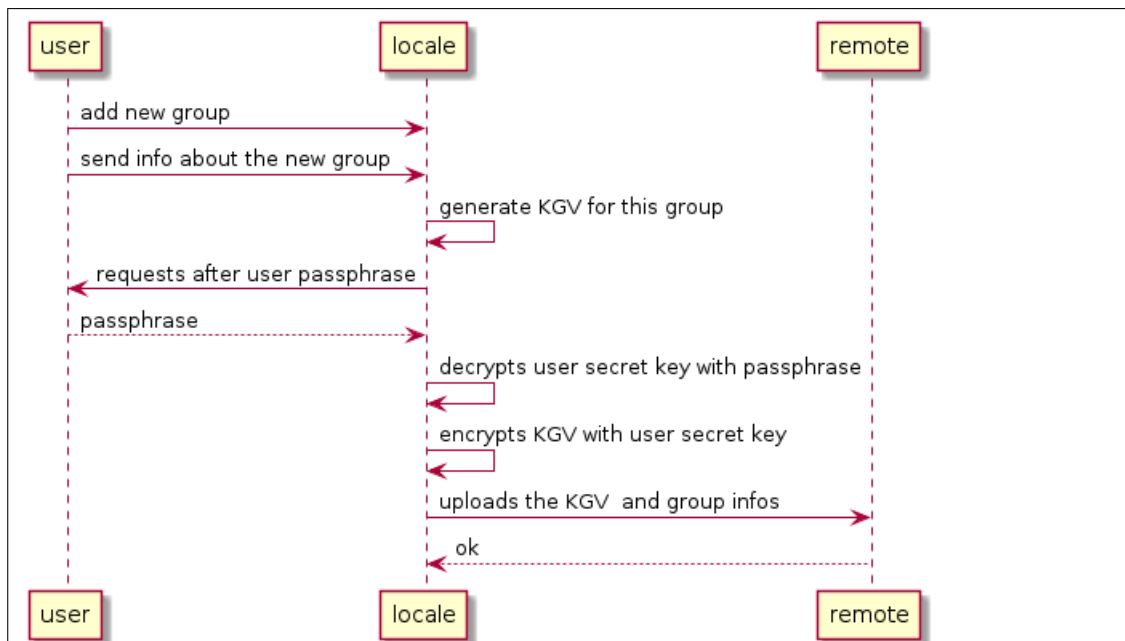


Abb. 4.15: Neue Gruppe anlegen

- Alice klickt auf eine neue Gruppe an und gibt den Name der neuen Gruppe im erscheinende Pop-up
- LocalServer lädt Alice öffentliche und private Schlüssel runter.
- Alice wird gefordert, Ihre Passphrase einzugeben (Falls noch nicht vorhanden)
- Es wird eine symmetrische Schlüssel (SGK)
- SGK wird mit Alice öffentliche Schlüssel verschlüsselt.
- SGK wird mit Alice private Schlüssel signiert.
- Nach die vorherige Vorgang, werden ein Gruppe Objekt und SymKey Object erzeugt. Die beide Objekte werden von LokalServer zu RemoteServer geschickt.

Kapitel 5

5 Implementierung und Evalierung

Bei diese Abschnitt geht es um die konkrete Implementierung von der verschiedenen Softwareteils, nämmlich :

- LocalServer
- RemoteServer
- CryptUtils
- Frontend
- Inbetriebnahme-programm

Es wird als erste eine Unterkapitel über die wesentliche Technologien die für die Anfertigung des Projektes benötigt wurde, gefolgt von deren Beschreibung. Insbesondere wird Wert auf die Funktionalität von die Framework gelegt, die eine bedeutende Rolle in der Implementierung haben, und die Gewährleistung von relevante Sicherheitmechanismen zur Erfüllung die vorgegebene Anforderungen.

5.1 Überblick auf die Technologie

	Programmiersprache	Technologie/Framework	Buildtools
CryptUtils	JAVA	JCE, Guava	Maven
RemoteServer	JAVA	Spring, Hibernate, Guava	Maven
LocalServer	JAVA	Spring, Guava	Maven
Frontend	JavaScript/HTML/CSS	AngularJS	GruntJS

Tab. 5.1: Technologie-Überblick

5.2 Allgemein Designentscheidungen

5.2.1 JSON-Format

System-weit wird JSON-Format bevorzugt um die Daten zwischen die verschiedene Softwarekomponente zu transpotieren. Explizit ausgedruckt, heißt es dass alle High-end

Funktionen bzw. die Funktion die durch eine Softwarekomponent zur Außenwelt verfügbar gemacht wurden exportieren Daten in JSON-Format. Diese Entscheidung lässt sich bei der Interoperabilität gründen, sowie auch Kriterien wie Einheitlichkeit von Softwareschnittstellen, was bei der Weiterentwicklung von grossen Bedeutung ist. Durch den Einsatz von JSON als Export-Format wird beispielsweise das Ersetzen von Softwarekomponente einfach. Beim Einsatz von JSON wurde die von Google entwickelte GSON Bibliothek benutzt.

5.2.2 UTF-8 und Base64

5.2.3 HTTP Headers

Headers sind mächtige Werkzeuge wenn es zum Internet kommt, und wichtiger noch im Bereich Security von Webbasierte Anwendungen. Bei der Entwurf von dieser Arbeit, wurde die Entscheidung getroffen soviel wie möglich auf die Standard zu halten, insbesondere bei sicherheitsrelevante Bereiche. Die richtige Einstellung/Konfiguration von manche Headers tragen wesentlich bei, um der Sicherheitsgrad eine Webanwendung zu erhöhen. Es wird nochmal über Headers die Rede sein, bei alle Softwareteil wo sie gesetzt bzw. geprüft werden (LocalServer, RemoteServer, Frontend), aber hier ist schon mal wichtig darüber zu erwähnen und eine gesamte Überblick über die Headers die Systemweit eingesetzt werden.

Headername	Wert	gesetzt bei	Laufzeit	
Content-Security-Policy	script-src 'self'	LocalServer	Erste Request an LocalServer	1
Authorization	SRP	LocalServer	Erste Login Request	2
WWW-Authenticate	SRP	LocalServer	Erste Login Request	3
realm	realm	LocalServer	Erste Login Request	4
hash-algorithm	SHA256	LocalServer	Erste Login Request	5
X-XSRF-TOKEN	=====	LocalServer	Erste Request an LocalServer	6
AUTH-TOKEN	=====	RemoteServer	Nach erfolgreichen Authentifizierung	7
EXPIRES-IN	=====	RemoteServer	Nach erfolgreichen Authentifizierung	8
Client-Public-Key	=====	LocalServer	Erste Login Request	9
Server-Public-Key	=====	RemoteServer	Erste Login Request	10
typ	JWT	RemoteServer	Nach erfolgreichen Login	11
alg	RS512	RemoteServer	Nach erfolgreichen Login	12

Tab. 5.2: Headers

- (1) Content-Security-Policy spielt eine bedeutende Rolle um XSS-Angriffe zu vermeiden. mit dem Wert script-src 'self' weist die Header hin, dass alle JavaScript source Datei nur von Server geladen werden dürfen. In unsere Fall vom LocalServer.

- (2) (3) (4) und (5) Informieren den Webbrowser über dem Authentication Algorithmus bzw. dem Hash-Algorithmus, der eingesetzt wird.
- (6) Wichtige Header gegen XSRF-Attacke (Cross Site Request Forgery)¹

5.3 Frontend

In diesem Kapitel handelt es sich um der Clientoberfläche in Form eine Webapplication, die der Endbenutzer auf irgendeinem Rechner der über eine Webbrowser verfügt aufrufen kann. Hier ist noch mal zu erinnern, dass eine der wichtige Anforderung von dieser Arbeit war das der Software so konzipiert wird, dass es kein zusätzliche Softwareinstallation benötigt wird. Das Webapplication wurde in Form eine sog. Single Page Application, wie bereits erwähnt, handelt es sich um eine Technik Webseite zu entwerfen, so dass die Bedingung von Webapplication ähnlich ist wie von Benutzer schon bekannt Computersprogramm. Neben der Usability und Portierungsargumente kommt auch die strenge Haltung von wichtigen Softwarearchikture und Regeln die mit Einsatz von AngularJS verbunden sind, nämlich Separation of Concern und MVVM . Ausserdem ermöglicht angular eine wesentliche Reduzierung von Request an Server, dass hat zur Folge dass der Man-in-the-middle ehe wenige Material bekommt um eine Offline-attacke durchzuführen.

AngularJS und Security

An der Webbrowser wird vorwiegend Angularjs eingesetzt. Was Sicherheit angeht, werden nämlich den Angular Speicher strategie, cookies , die über den angular-service einsetzbar ist.

Was path und domain angeht wurden die Default Werte gelassen, und zwar Cookie steht zur Verfügung für aktuelle Pfad und alle untergeordnete Pfade bzw. Cookie steht zur Verfügung nur für die Application domain. Was hier konfiguriert wurde war den Parameter secure mit den wert true und expires mit eine Date Instance in Form eine Zeichenkette, der konfigurierte die Lebensdauer der Cookies. Zusammengefasst darf nur der aktuelle Angular Application auf der Cookies zugreifen, und der Lebensdauer von der Cookie wird von Angular managiert. in der Abbildung ?? wird die in Cookie gespeicherte Daten nach Sechs Minute zerstört, infolgedessen der Benutzer automatisch ausgeloggt wird. die Ablaufzeit wird von der Server bestimmt und über der Header „EXPIRES_IN“ zur LocalServer bzw. zur Angular Applikation übermittelt.

¹XSRF : Manipulation von Parametern, so dass im Browser Skriptcode ausgeführt wird; häufig auch synonym zu HTML-Injection verwendet[9]

```

1  var options =
3  {
    secure : true
5  };

7  function refresh_storage(){
    var d = new Date( Auth.getHeader( EXPIRES_TOKEN ) );
    var n = d.toUTCString().toString();
    options.expires = n;
11 }

```

Abb. 5.1: Angular: Cookie Konfigurierung

5.3.1 Ausstattung von Routes

Die unterstehende Grafik repräsentiert die mögliche Aktionen, die den Benutzer mithilfe der Webbrowser auslösen kann. Es lässt sich dadurch nochmal eine graphische abstrahierende Darstellung von funktionale Anforderungen darstellen.

5.4 allgemeine Implementierung Entscheidung von Local-und Remoteserver

Beide LocalServer und Remoteserver wurde in Java mithilfe der Framework Spring programmiert. Spring ist eine allgemeine Zweck Framework um Desktop- sowie Internetbasierte Anwendungen zu implementieren. Folgedessen ist Spring meistens eng mit Internet bezogene Context verbunden, was das Testen von Software behindert oder sogar nicht möglich macht.

Abb. 5.2: Routes-Übersicht

5.5 LocalServer

LocalServer wurde konzipierte um in ein USB-Stick lauffähig zu sein, und zwar ganz ununterschieden von Betriebssystemart². Das Ziel wurde erreicht indem dass LocalServer mit Java programmiert wurde. Das Programm an sich und eine lauffähige JVM liegen dann in der USB Stick.

²Jetzige Stand von LocalServer wurde auf Linux basierte Betriebssystem und Window Betriebssystem erfolgreich getestet.

Auch wenn LocalServer mit einem Plattform unabhängige Programmiersprache implementiert wurde, musste nochmal das Programm rücksichtvoll konfiguriert werden, damit beispielsweise Der LocalServer nicht auf ein Port zuweisen den schon für ein andere Zweck auf der Benutzerrechner belegt wurde.

LocalServer Module

CryptoUtils

Diese Module enthält alle kryptographische Funktionalitäten, welche für diese Arbeit eine Rolle spielen, nämlich :

1. Symmetrische Verschlüsselung und symmetrische Schlüssel-Generator.
2. Asymmetrische Verschlüsselung und asymmetrische Schlüssel-Generator
3. Passwort basierte asymmetrische Schlüssel-Generator
4. Digitale Signature
5. Hashwert-Berechnung
6. Zufall-Generator (Salt-Generator)

Alle oben genannte Funktionalität wurden als Funktionsmodule (Gedächtnislose-Module) implementiert. Schlüssellänge und eingesetzte Algorithmus können durch die Konfigurationsdatei geändert werden.

Die obengennante Untermodule verfügen über eine einheitliche Schnittstelle, Parameter-Typ und Rückgabewert sind immer Base-64 codierte JSON-Zeichenkette.

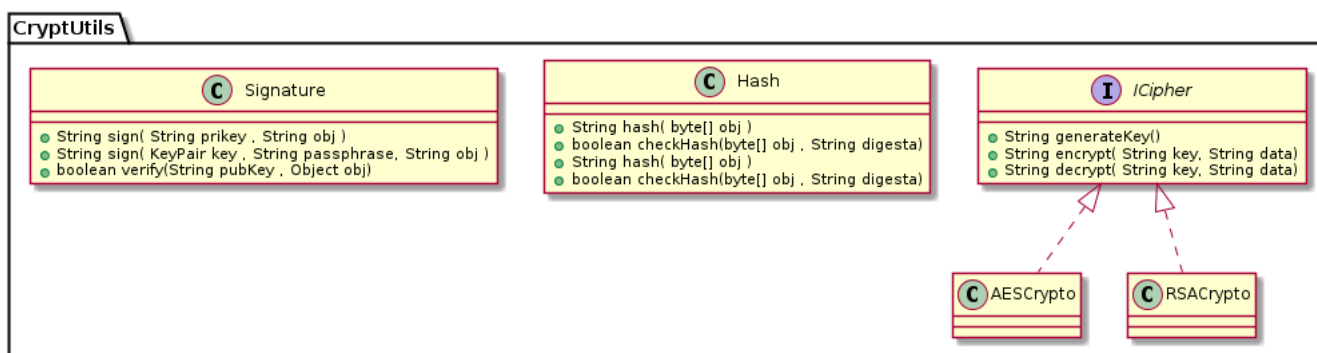


Abb. 5.3: CryptoUtils Module

Alle Klasse werden sowie in der oben stehenden Abbildung weiter im LocalServer benutzt („AS IS“). Ausnahmeweise wird AESCrypto abstrahiert um eine Spezialisierung für Dateiverschlüsselung zu entwerfen.

Konfigurationsdatei

```
1 app:
   name: CryptLocal
3   description: CryptoLocal Server
   server:
5     port: 0
   multipart:
7     maxFileSize: 128KB
     maxRequestSize: 128KB
9   remoteserver:
     uri:
11     register: /session/register
     logout: /session/logout
13     challenge: /session/login/challenge
     authenticate: /session/login/authenticate
15     documents: /api/groups/{prefix}/documents/{suffix2}
     friends: /api/{prefix1}/friends/{suffix1}/{suffix2}
17     users: /api/{prefix1}/{prefix2}/users/{suffix1}
     groups: /api/groups/{suffix1}/{suffix2}/{suffix3}
19     keypairs: /api/{userId}/keypair
     symkeys: /api/{groupId}/keysym
21     url: http://localhost:8080/
```

Listing 5.1: application.yml

Abb. 5.4: Konfigurationsdatei

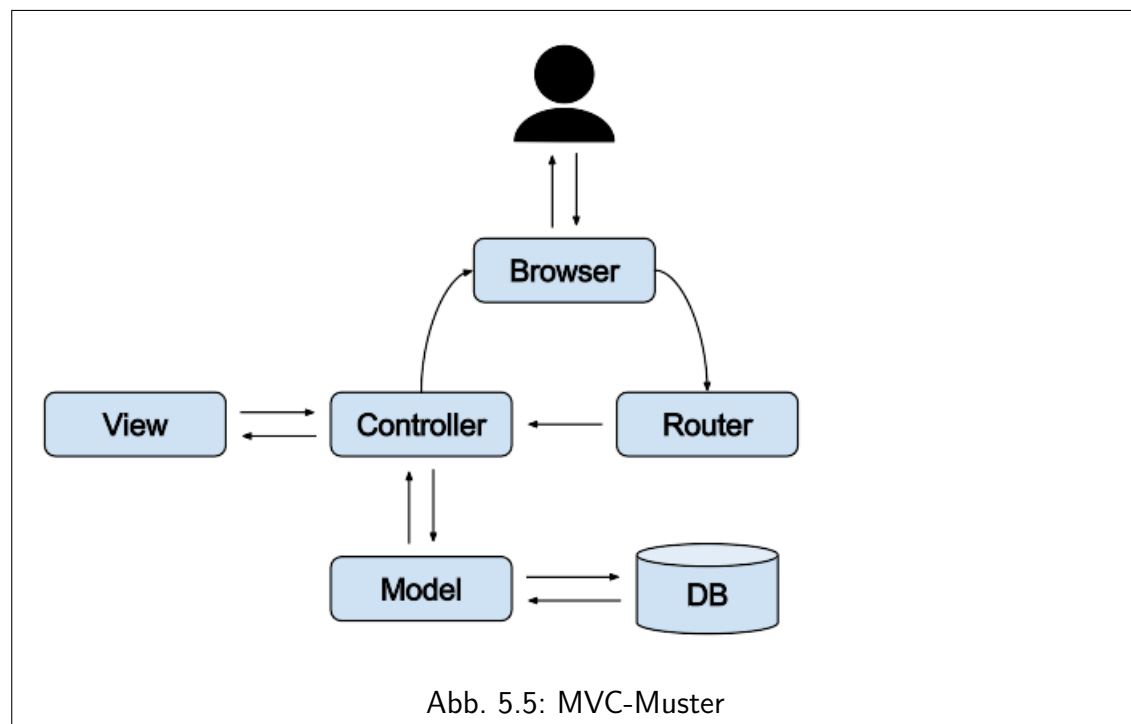
Abbildung 5.5 zeigt eine exemplarische Spring-Boot Projekt Konfiguration wie es in den Projekt eingesetzt wurde. Spring-Boot übernimmt die Verwaltung von Konfigurationsdatei und sorgt zur Verbindung von in der Konfigurationsdatei eingetragene Wert und dessen Verweis im Java Code, oder auch zur Bestimmung der Kontext. in der oben benannte Konfigurationsdatei wird zum Beispiel der Wert Port zu 0 gesetzt, was dafür sorgt dass LocalServer immer eine beim Clientrechner nicht verwendete Port benutzt. Beim RemoteServer muss dieser Wert jedoch immer fest sein. Weitere werden in Konfigurationsdatei Werte wie Schlüssellänge, Servermessage und Datenbankverbindungsinformationen festgelegt. Es ist anzudeuten dass die Konfigurationswert von RemoteServer modifiziert sein können um etwa die Datenbankverbindungsinformationen auf der RemoteServer anzupassen. Diese Modifikation geschieht durch Modifizierung der Konfigurationsdatei oder per Commandline-Argument beim aufrufen von Java zur Kompilierung.

```
$ java -jar -Dspring.profiles.active=production CryptoRemote-0.0.1-SNAPSHOT.jar
```

Model View Controller

Model view controller (MVC, englisch für Modell Präsentation Steuerung) ist ein Muster zur Strukturierung von Software-Entwicklung in die drei Einheiten Datenmodell, Präsentation und Programmsteuerung. Ziel des Musters ist ein flexibler Programmentwurf, der eine spätere Änderung oder Erweiterung erleichtert und eine Wiederverwendbarkeit.

Diese Entwurf ist ein Kern-feature von Spring-boot, die verschiedene Erleichterung bereitstellt für eine einfache Implementierung von Software nach MVC Modell. Diese Erleichterungen sind zum Beispiel Annotations wie „@RestController“ „@Service“ oder auch „@Component“.



Der Controller (mit RestController oder Controller Annotation vorgesehen) ist mit Applikation-Logik (mit Service Annotation vorgesehen) verkoppelt. Controller-Einheit stellt auch die Ressource zur Außenwelt zur Verfügung.

Applikation-Logik besteht aus zwei Schichten, Controller und Service. Controller die mit Internet-spezifische Technologie gekoppelt ist nimmt die Anfragen entgegen. Bearbeitung der Anfrage an sich geschieht im Service. Dank diese Schichtmodell und Entkopplung von Service und Controller wird erzielt, dass die Integrationstest der Software ohne Http-spezifische Softwarekomponente durchgeführt werden kann.

RestClient

RestClient Module ist dafür verantwortlich Http-Anfrage an RemoteServer versenden.

Fehlerbehandlung und Benutzerrückmeldung

Fehlerbehandlung ist der Kern ein Softwaresystem. Implizite Anforderungen beim Entwurf eines Software ist immer eine sinnvolle Rückmeldung an Benutzer (Aktion erfolgreich bzw. nicht erfolgreich). Die Gründe für eine nicht erfolgreiche Aktion könnten an viele Stelle sein:

- Falsche Request
- momentane nicht ansprechbare Resource
- Fehler im Code

Je nach Fehlerart bzw. Meldungsart, ist es wichtig den Benutzer ein gewisse Feedback zu geben und schliesslich den Fehler behandeln können oder dokumentiert für eine nachtragliche Verbesserung des Softwares

Anhand dieser Arbeit wurde folgende Http-Status code benutzt :

	Status-code	Kurzbeschreibung	Beschreibung
1	200	OK	
2	201	CREATED	
3	202	DELETED	
4	204	No Content	
5	401	Error Authentication	
6	403	Forbidden	
7	405	Not allow	
8	400	ERROR	

Abb. 5.6: Http-Status code. RFC2616[2]

Mit Hilfe von Angular Http Interceptor wurde ein Logik implementiert, die die Status-code von Http Response filtert, und jenach status code eine zugehörige Rückmeldung im Browser anzeigt.

Sollte ein Fehler Code auftauchen, dann wird automatische eine Antwort mit Status-Code 500 an Browser geschickt. Diese Sachverhalten ist aber nicht gewünscht, da es Benutzer und/oder ein eventueller Angreifer darüber informiert dass beispielsweise ein NullPointerException in Code aufgetaucht wurde. Dies kann der Angreifer benutzen um das System zu hacken. RemoteServer und LocalServer haben ein Filter, der die Antwort an Client filtert und 500 Status code durch 400 Status code ersetzt, weitere wird die verbose Nachricht was die Fehlerquelle beschreibt durch eine weniger informative Message ersetzt. Alle Code Fehler, materialisiert durch 500 Status Code werden dokumentiert, in Log Datenbank und Log-Datei, ein Email mit genau Beschreibung des Fehlers wird zur Admin geschickt jedes mal das solche Fehler auftauchen.

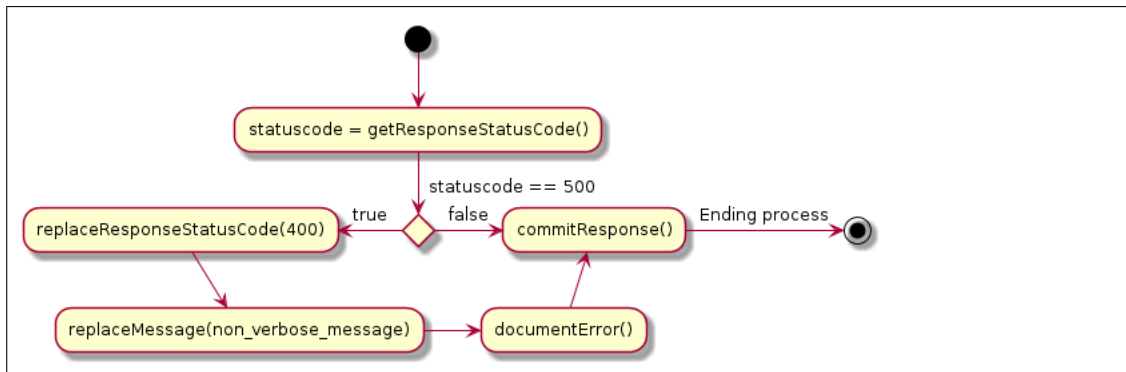


Abb. 5.7: Status-Code Filter

Caching

Um häufigere Http-Anfrage an RemoteServer zu vermeiden, wurde eine Caching-Strategie im LocalServer und Angular Frontend-Application entworfen. Ein Vorteil von Caching ist es die Overhead zu vermeiden. Und eventuelle ein Dritter der die Kommunikation abhört, nicht durch häufige Anfrage mit Infos füttert. Caching spielt auch eine bedeutende Rolle bei Skalierung von Software und führt dass die Software mehr Fehlertolerant wird. Caching erbringt mehr Leistung durch eine Verringerung der durchschnittlichen Latenzzeit von einer Reihe von Interaktion und erhöht die benutzerwahrgenommene Leistung.

```

@Configuration
@EnableCaching
public class CacheConfig implements CachingConfigurer {

    public final static String CACHE_USERS = "cacheUsers";
    public final static String CACHE_DOCUMENTS = "cacheDocuments";
    public final static String CACHE_GROUPS = "cacheGroups";
    public final static String CACHE_FRIENDS = "cacheFriends";
    public final static String CACHE_SESSION = "cacheSession";
}
  
```

Abb. 5.8: Caching configuration

Weitere wird Caching benutzt um Session Daten wie zum Beispiel chiffriert Passphrase zu speichern. Dadurch wird verhindert dass Benutzer immer wieder seine Passphrase eingeben muss und eventuelle Keylogger-attacke vermieden.

Datenbank Schema

Das unterstehende Abbildung zeigt die Datenbankschema. Dabei ist zu beachten dass alle als kritischen bzw. sensiblen Informationen mit zwei wichtige Einträge vorgesehen sind und zwar : „hash“ und „signature“. Wie vorher erwartet, sind diese Einträge wichtig für die Integritätsüberprüfung von Daten.

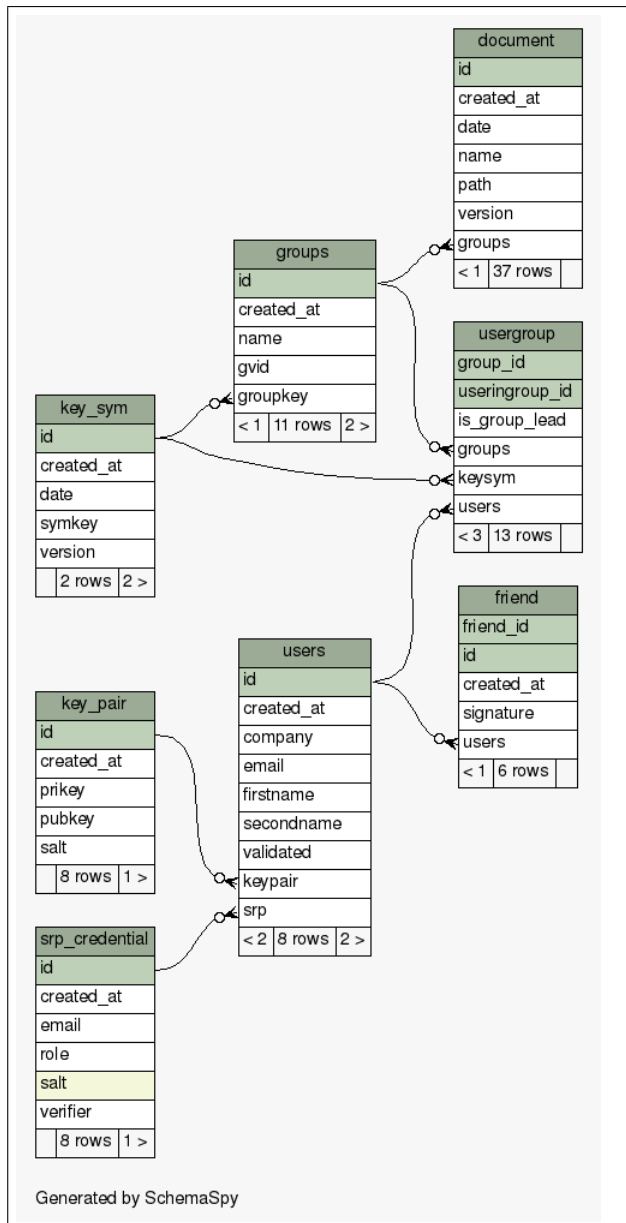


Abb. 5.9: Datenbank Schema

Datenbankschema beschreibung :

Users: Im Schemas Users wird die Benutzerinformationen gespeichert

SrpCredentials: In dieser Tabelle werden Benutzerinformations bezüglich SRP-A, aus Sicherheitsgründe wurde diese Informationen vom Users Schema entkoppelt.

Friends: Friends ist die Materialsierung von eine Vertrauenbeziehung zwischen zwei Benutzer. stellt die Vertrauensbeziehung zwischen dem Benutzer dar. Sobald ein Benutzer den öffentlichen Schlüssel eines anderen Benutzers signiert, wird eine Zeile in dieser Tabelle eingefügt, wobei die signority_id und user_id die Id der vertrauenden und des Vertrauten sind.

Roles: Benutzer Roles, die sind festgelegt, wert für eine Role könnten : „Admin“ oder

„User“ sein. Es ist zu beachten dass pro Deployment nur ein einziger Benutzer als Admin markiert werden darf. Weitere Benutzer werden alle als „user“ markiert.

Session: Es handelt sich um die Materialisierung einer aktiven Session

UsersGroups: Überbrücke zwischen die Tabelle User und Group

Groups: Hier werden die Informationen relativ zu den Gruppen gespeichert. Hier ist der Attribute SGK ist der gemeinsam Gruppe Schlüssel und ist mit dem öffentlichen Schlüssel des Administrators oder Gründer dieser Gruppe verschlüsselt.

Documents: beinhaltet Dokumenteninformationen.

SymKeys: Symmetrischen Schlüssel

PairKeys: Assymetrischen Schlüssel.

5.6 Evaluierung

In diesem Abschnitt wird überprüft, in wieweit die Umsetzung des Lösungskonzeptes die geforderten Anforderungen an der Cryptone-System erfüllt. Zusätzlich wird auf Maßnahmen eingegangen, um Korrektheit des Dienstes sicherzustellen.

5.6.1 Anforderungserfüllung

Die Anforderungen aus Kapitel 3 wurde in in funktionale und nichtfunktionale Anforderungen unterteilt. Während die funktionalen Anforderungen durch konkrete softwaretechnische Einheit (Klasse, Module) werden konnten, beziehen sich die nicht-funktionalen Anforderungen auf Eigenschaften des Gesamtsystems, die sich nicht an einzelnen Modulen festmachen lassen.

Um die funktionalen Anforderungen mit den Modulen abzugleichen, mussten die einzelnen Anforderungen teilweise in kleinere Schritte unterteilt werden, um Software besser zu gestalten.

Die Kernfunktionalität von Software (Authentication, Registrierung, Dateiaustauschoptionen ...) wurde durch eine Unit-Test Programm getestet. [Siehe Anhang für die Ergebnisse], sowie auch nicht-funktionale Anforderungen wie :

- Funktionparameter Integrität prüfung (Null-Object , nicht genug Argument, falsche Argument, Argumenttyp Prüfung)
- Allgemein Laufzeit Fehler (Zum beispiel Cryptographie-Algorithmus nicht Verfügbar)

	Nach Kompilierung	Betriebssystem	Laufumgebung
LocalServer	JAR	unabhängig	JRE
RemoteServer	WAR	unabhängig	Servlet Container
FrontenApp	JS, HTML	unabhängig	Webbrowser

Abb. 5.10: Cryptone Compilierung

Nicht funktionale Anforderungen : Portabilität

RemoteServer und LocalServer sollte portable sein.

die obere Tabelle, zeigt die Verschiede Softwareteile des Projekts, Alle Softwareteil können ganz unabhängig von Betriebssystem zur Laufen gebracht werden. Obwohl Die Softwareteilen betriebssystem-webbrowser bzw. laufumgebungsoftware unabhängig sind, müssen sie eventuelle angepasst oder zusätzliche Software wie Browser (Internet Explorer hat eine renomierte Compatibiltät-Problem mit den aktuellten Internet Standard) installiert werden, aber keinerlei wird eine Softwareänderung benötigt.

Stand des Tests:

- LocalServer läuft auf eine USB-Stick : diese USB-Stick ist mit Java Runtime Environment (JRE) (64 und 32 Bit) vorgesehen, sowie Windows JRE (64 Bits), und eine skript (bash für Linux System und bat für Window), schliesslich das LocalServer Software an sich. LocalServer wurde erfolgreich auf Ubuntu 15.10 wily 64 Bits und Windows 7 64 Bits getestet.
- RemoteServer wurde erfolgreich unter Ubuntu 15.10 wily 64 Bits, mit Apache Tomcat 8.x getestet
- FrontendApp wurde erfolgreich unter Ubuntu 15.10 wily 64 Bits und Windows 7 64 Bits mit Webbrowser Google Chrome und Mozilla Firefox erfolgreich getestet.

Nicht funktionale Anforderungen : Usability

Trotz die Komplexität des Projektes wurde die CryptoSystem-Client User-Freundlich wie möglich entworfen. Die Interaktion geschieht durch eine Webseite. Die mögliche Aktionen („sign-in“, „sign-up“, „upload“, „download“, „share“, „add-friend“), Drag-and-Drop sind üblich und von Benutzer aus andere Webseite bekannt. Fakt ist dass Zielbenutzer von Cryptone-System sind mit diese Aktionen schon gewöhnt. Weitere alle Komplexe Logik passiert im LocalServer und der Endbenutzer kriegt nichts mit von dieser Logik.

LocalServer kümmert sich auch selbst darum eine freie nicht verwendete Port zu belegen und starte auch die Webseite (Loginseite von LocalServer).

Was von Benutzer verlangt ist sein USB-Stick anzuschliessen und auf Start-Skript (BAT für Window und BASH für Linux) zu starten.

Nicht funktionale Anforderungen : Functionality und Maintainability

- Interoperability und Compliance, LocalServer und RemoteServer sind nach REST-Full Standard entwickelt wurde. Überall Daten die produziert werden sind in JSON-Format. LocalServer oder RemoteServer können getausch werden oder in eine andere Programmiersprache programmiert solange die Ersatzsoftware die REST-Spezifikation erfüllt und auf schon definiert Resource mit definierten Methoden zugreift.

6 Kapitel 6 **Zusammenfassung und Ausblick**

6.1 Zusammenfassung

KryptoOne-System ist ein Internet-basierte Dokumentverwaltung (Dokumentaustausch, Dokumentablagerung ..) konzipiert mit einer intuitiven Benutzer Interaktion durch Webbrowser und mit dem Ziel, kryptographische Komplexität zu abstrahieren, das heisst so gut wie keine Kryptographie-relevante Aufgabe an Benutzer zu überlassen. Im Unterschied zu anderen Cloud-Services werden die Dateien verschlüsselt, bevor sie hoch geladen werden. KryptoOne-System verwaltet auch die Benutzer und wendet das Konzept von Vertrauensbeziehung im Dateiaustausch an. Ferner wird die Sicherheits-Stärke den Unternehmen, welche die Software benutzen, überlassen. Das heißt, ein Unternehmen ist nicht durch einen beschränkten Sicherheitsgrad gehandikapt.

Die bisherige Lösung diese Problems sah einen großen Aufwand für Endbenutzer vor. Benutzer sollte sich selber darum kümmern, Schlüssel zu erzeugen, Schlüssel zu managen und Schlüssel an Kommunikationspartner zu übergeben. Die entscheidenden Nachteile dabei waren der Aufwand des oben beschriebenen Prozesses, die Voraussetzung, dass die Benutzer sich mit Kryptographie auskennen, die Schlüsselverteilungsproblematik und schließlich die Installation von neuer Software.

Zu Beginn der Arbeit wurden die Probleme der bisher eingesetzten Lösungen genauer beleuchtet und die Motivation für eine Komplexitäts-transparente Cryptosoftware mit spezifizierten Anforderungen abgeleitet. Anschließend wurden die Ziele der Arbeit formuliert.

Im theoretischen Teil wurde auf die zur Umsetzung des CryptoOne-Systems benötigten Grundlagen eingegangen. Zuerst wurden die heutigen sicheren kryptographischen Standards und wichtige Begriffe erläutert. Die Einschränkungen und Nachteile dieser Standards wurden beleuchtet. Im Anschluss wurde darauf eingegangen, wie sich durch das Zusammenspiel dieser Standards die besagten Einschränkungen umgehen lassen. Ziel dabei war es, den Sicherheitsgrad der entworfenen Lösung zu erhöhen, wie zum Beispiel Hybrid-Verschlüsselung aus RSA und AES, um eine höhere Performance zu erreichen.

Der praktische Teil dieser Arbeit wurde in vier Kapitel gegliedert: Anforderungs-Analyse, Erstellung eines Lösungskonzeptes, Umsetzung und Bewertung des umgesetzten Lösungskonzeptes.

Durch die Anforderungs-Analyse wurden die konkreten Anforderungen an das CryptoOne-System herausgearbeitet. Dabei wurden sowohl funktionale, als auch nicht-funktionale Anforderungen berücksichtigt.

6.2 Ausblick

Programm-Listings

5.1	application.yml	42
-----	---------------------------	----

Tabellenverzeichnis

Literaturverzeichnis

- [1] ISO/IEC: *Information technology - Software Product Evaluation - Quality characteristics and guidelines for their use*. <http://www.cse.dcu.ie/essiscope/sm2/9126ref.html>. Version: 2016. – [Online; Stand 14. maez 2016]
- [2] IETF: *Hypertext Transfer Protocol – HTTP/1.1*. <https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>. Version: 2016. – [Online; Stand 14. maez 2016]
- [3] DUDEN: *das Passwort*. <http://www.duden.de/rechtschreibung/Passwort>. Version: 2016. – [Online; Stand 14. maez 2016]
- [4] POSTEL, J.: *FILE TRANSFERT PROTOCOL (FTP)*. <https://tools.ietf.org/html/rfc959>. Version: 2016. – [Online; Stand 14. maez 2016]
- [5] COOPER, M. ; DZAMBASOW, Y. ; HESSE, P. ; JOSEPH, S. ; NICHOLAS, R.: *Internet X.509 Public Key Infrastructure: Certification Path Building / RFC Editor*. Version: September 2005. <http://www.rfc-editor.org/rfc/rfc4158.txt>. RFC Editor, September 2005 (4158). – RFC. – ISSN 2070–1721
- [6] WU, T.: *The SRP Authentication and Key Exchange System*. <https://www.ietf.org/rfc/rfc2945.txt>. Version: 2016. – [Online; Stand 14. maez 2016]
- [7] IETF: *JSON Web Signature (JWS)*. <https://tools.ietf.org/html/rfc7515>. Version: 2016. – [Online; Stand 14. maez 2016]
- [8] IETF: *JSON Web Encryption (JWE)*. <https://tools.ietf.org/html/rfc7515>. Version: 2016. – [Online; Stand 14. maez 2016]
- [9] INFORMATIONSTECHNIK, BSI: B. d.: *Sicherheit von Webanwendungen, Maßnahmenkatalog und Best Practices*. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/WebSec/WebSec_pdf.pdf?__blob=publicationFile. Version: 2016. – [Online; Stand 14. maez 2016]
- [10] IETF: *JSON Web Token (JWT)*. <https://tools.ietf.org/html/rfc7519>. Version: 2016. – [Online; Stand 14. maez 2016]
- [11] *SRP Secure Remote Password Protocol*. <http://tools.ietf.org/html/rfc2945>, . – Accessed: 12-12-2015, snapshot: reference/ietf/rfc2945
- [12] *GSON JSON Manipulation Framework*. <http://google.com/gson>, . – Accessed: 12-12-2015, snapshot: reference/google/gson

- [13] JCA Java Cryptography Architectur reference guide. http://java.com/jca/jca_reference_guide, . – Accessed: 12-12-2015, snapshot: reference/java/gson
- [14] GERNOT STARKE, Karl Eilebrecht und: *Patterns kompakt: Entwurf für effektive Software-Entwicklung*. Bd. 4. London : Springer Vieweg. – 34–38 S.
- [15] ZIMMERMANN, P.: The Official PGP User's Guide. In: *MIT Press* (1995), Mai
- [16] ATKINS, Derek ; STALLINGS, William ; ZIMMERMANN, Philip: PGP Message Exchange Formats / RFC Editor. Version: August 1996. <http://www.rfc-editor.org/rfc/rfc1991.txt>. RFC Editor, August 1996 (1991). – RFC. – ISSN 2070–1721
- [17] J. FEIGENBAUM, J. L. M. Blaze B. M. Blaze: Decentralized Trust Management. In: *Proceedings of the IEEE Conference on Security and Privacy. Oakland* (1996), Mai
- [18] STALLINGS, W.: The PGP Web of Trust. In: *BYTE* (1995), February
- [19] ABDUL-RAHMAN, Alfarez: The PGP Trust Model. In: *Department of Computer-Science* (1995), Februar
- [20] C., Adams ; S., Lloyd: Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations. In: *Macmillan TechnicalPublishing* (1999)
- [21] HOUSLEY, R. ; POLK, W. ; FORD, W. ; SOLO, D.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile / RFC Editor. Version: April 2002. <http://www.rfc-editor.org/rfc/rfc3280.txt>. RFC Editor, April 2002 (3280). – RFC. – ISSN 2070–1721
- [22] DAVID W. CHADWICK, Alexander O.: *The PERMIS X.509 role based privilege management infrastructure*. Bd. 9. 2. Original Research Article Future Generation Computer Systems, 2003. – 277–289 S.
- [23] CHADWICK D.W., Ball E. Otenko A. A. Otenko A.: Role-based access control with X.509 attribute certificates. In: *IEEE Internet Computing* (2003), März
- [24] ELLISON, Carl M. ; FRANTZ, Bill ; LAMPSON, Butler ; RIVEST, Ron ; THOMAS, Brian ; YLONEN, Tatu: SPKI Certificate Theory / RFC Editor. Version: September 1999. <http://www.rfc-editor.org/rfc/rfc2693.txt>. RFC Editor, September 1999 (2693). – RFC. – ISSN 2070–1721
- [25] SAITO T., Okuno H. Umesawa K. K. Umesawa K.: Privacy-enhanced access control by SPKI and its application to Web server. In: *Enabling Technologies: Infrastructure for Collaborative Enterprises* (2000), Juni

- [26] CLARKE, Dwaine ; ELIEN, Jean-Emile ; ELLISON, Carl ; FREDETTE, Matt ; MORCOS, Alexander ; RIVEST, Ronald L.: *Certificate Chain Discovery in SPKI/SDSI*. September 2001
- [27] HALPERN, Joseph Y. ; MEYDEN, Ron Van D.: A Logic for SDSI's Linked Local Name Spaces. In: *In Proc. 12th IEEE Computer Security Foundations Workshop*, 1999, S. 111–122
- [28] BLAZE, Matt ; FEIGENBAUM, Joan ; IOANNIDIS, John ; KEROMYTIS, Angelos D.: The KeyNote Trust-Management System Version 2 / RFC Editor. Version: September 1999. <http://www.rfc-editor.org/rfc/rfc2704.txt>. RFC Editor, September 1999 (2704). – RFC. – ISSN 2070–1721
- [29] IOANNIDIS, Matt Blaze J. ; KEROMYTIS, Angelos D.: Trust Management for IPsec. In: *In Proceedings of the Internet Society Symposium on Network and Distributed Systems Security (SNDSS) 2001* (2001), Februar, S. 139 – 151
- [30] KEROMYTIS, Matt Blaze John Ioannidis Angelos D.: KeyNote: Trust Management for Public-Key Infrastructures. In: *In Proceedings of the 1998 Security Protocols International Workshop, Springer LNCS 1* (1998), April, S. 59 – 63
- [31] A.BUCHMANN, Johannes ; KARATSIOLIS, Evangelos ; WIESMAIER, Alexander: Private Keys. Version: 2013. http://dx.doi.org/10.1007/978-3-642-40657-7_4. In: *Introduction to Public Key Infrastructures*. Springer Berlin Heidelberg, 2013. – ISBN 978–3–642–40656–0, 61–74
- [32] MORIARTY, K. ; NYSTROM, M. ; PARKINSON, S. ; RUSCH, A. ; SCOTT, M.: PKCS 12: Personal Information Exchange Syntax v1.1 / RFC Editor. Version: July 2014. <http://www.rfc-editor.org/rfc/rfc7292.txt>. RFC Editor, July 2014 (7292). – RFC. – ISSN 2070–1721. – <http://www.rfc-editor.org/rfc/rfc7292.txt>
- [33] KALISKI, Burt: PKCS 1: RSA Encryption Version 1.5 / RFC Editor. Version: March 1998. <http://www.rfc-editor.org/rfc/rfc2313.txt>. RFC Editor, March 1998 (2313). – RFC. – ISSN 2070–1721
- [34] WIKIPEDIA: *RSA (cryptosystem)* — *Wikipedia, The Free Encyclopedia*. [http://en.wikipedia.org/w/index.php?title=RSA-20\(cryptosystem\)-oldid=668601859](http://en.wikipedia.org/w/index.php?title=RSA-20(cryptosystem)-oldid=668601859). Version: 2015
- [35] WIKIPEDIA: *Advanced Encryption Standard* — *Wikipedia, The Free Encyclopedia*. <http://en.wikipedia.org/w/index.php?title=Advanced-20Encryption-20Standard-oldid=666183137>. Version: 2015
- [36] SCHAAD, J. ; HOUSLEY, R.: *Advanced Encryption Standard (AES) Key Wrap Algorithm*. Internet Requests for Comments. Version: September 2002. <http://www.rfc-editor.org/rfc/rfc3394.txt>

- [37] NAFI, Kawser W. ; HOQUE, Tonny Shekha Kar Sayed A. ; HASHEM, M. M. A.: A Newer User Authentication, File encryption and Distributed Server Based Cloud Computing Security Architecture. In: *International Journal of Advanced Computer Science and Applications (IJACSA)* 3 (2012), S. 181–186. – ISSN 2156–5570