Sec^2 – Ein mobiles Nutzer-kontrolliertes Sicherheitskonzept für Cloud-Storage

Christopher Meyer[1] · Juraj Somorovsky[1] · Benedikt Driessen[1] · Jörg Schwenk[1] · Thang Tran[2] · Christian Wietfeld[2]

[1]Ruhr-Universität Bochum
{christopher.meyer, juraj.somorovsky, benedikt.driessen, joerg.schwenk}@rub.de

[2]Technische Universität Dortmund
{thang.tran, christian.wietfeld}@tu-dortmund.de

Zusammenfassung

Die fortschreitende Nutzung und Beliebtheit des Cloud Computings bringt neue Herausforderungen für sicherheitsrelevante Technologien mit sich - Dienstenutzer müssen sich lückenlos auf die Sicherheit der verwendeten Systeme verlassen können. Dieser Grundsatz schließt eine Sicherheitsarchitektur basierend auf dem bisherigen Vertrauensmodell zwischen Nutzer und Diensteanbieter aus.

Ziel des mit diesem Papier verbundenen Forschungsprojektes ist die Schaffung eines sicheren Verschlüsselungskonzepts für Cloud Computing, ohne dabei auf gängige Vertrauensmodelle zwischen Speicherplatz-Anbieter und Nutzer zurückzugreifen. Dabei soll die vollständige Kontrolle über die Daten auf der Seite der Nutzer liegen, wobei hingegen der Cloud Storage Provider keinerlei Kenntnis vom verwendeten Schlüssel oder den Klartext-Daten hat.

Darüber hinaus soll die Architektur kollaboratives Arbeiten in einem sicheren Umfeld fördern und zugleich eine nahtlose Datenkommunikation über ein heterogenes Netz ermöglichen. Die gesamte Architektur ist für die sichere Gruppennutzung ausgelegt und adressiert Kernprobleme der Nutzung von Online-Speicherdiensten:

- sichere Datenablage auf beliebigen Cloud basierten Speicherdiensten
- sichere Etablierung einer Gruppenkommunikation
- vollständige Kontrolle über die abgelegten Daten duch die Gruppenteilnehmer
- keine Vertrauensbeziehungen zwischen Nutzern und Diensteanbietern

Die vorgeschlagene Lösung resultiert aus dem Forschungsprojekt Sec^2, einem Kooperationsprojekt der Ruhr-Universität Bochum, TU Dortmund, sowie den beiden Industriepartnern Utimaco Safeware AG und Adesso Mobile Solutions. Das Projekt wird gefördert vom Bundesministerium für Bildung und Forschung (FKZ: *01BY1030*) und unterliegt der Leitung durch das Deutsche Zentrum für Luft- und Raumfahrttechnik.

1 Einführung

Durch die zunehmende Mobilität – sowohl von Endgeräten als auch von Diensten – ändern sich die Anforderungen an die Speicherung von Daten. Konnte man bei Desktop-Applikationen die eigenen Daten noch durch Personal Firewalls und Rechtevergabe sichern, so wurde dies im Client-Server-Paradigma schon schwieriger. Für das neue Mobile Cloud Computing im heterogenen Netz existieren so gut wie keine Schutzkonzepte: Daten können hier sowohl auf Client-Seite (mobile Endgeräte) als auch auf Serverseite (Web Services) auf beliebigen Geräten an be-

liebigen Orten verarbeitet werden. Schutzmaßnahmen, die auf ein Gerät (Antiviren-Software), einen Ort (Firewalls) oder ein Betriebssystem maßgeschneidert sind, greifen nicht mehr. Auch die SSL-Verschlüsselung lässt sich hier nur begrenzt einsetzen, da die Daten nur während des Transports geschützt und danach unverschlüsselt in der Cloud gehalten werden.

In Anbetracht des aktuellen Wandels gewinnt der Aspekt der Datensicherheit von Cloud basierten Diensten zunehmend an Relevanz [Brod08], welche von Nutzern auch bewusst wahrgenommen wird [Bök11]. Trotz dieses Sicherheitsbewusstseins existieren derzeit keine nutzerfokussierten Sicherheitslösungen. Bestehende Ansätze zielen auf die Verschlüsselung der Nutzdaten beim Cloud Service Anbieter. Dienste, die das Schutzziel Vertraulichkeit für in der Cloud gespeicherte Daten anbieten, verschlüsseln Dateien auf den dazu vorgesehenen Cloud Storage Servern und speichern sie nach der Verschlüsselung dort ab. Ein Beispiel eines solchen Dienstes ist Dropbox [Drop11], welcher Daten verwalten und verschlüsselt in Amazon S3 [LLC11] ablegen kann. Die Dropbox-Nutzer können damit auf der einen Seite sicher sein, dass die Vertraulichkeit ihrer Daten nach dem Verschlüsselungsvorgang gewährleistet ist, solange der Schlüssel niemand anderem bekannt ist, auf der anderen Seite lagen allerdings die Daten dem Diensteanbieter vor der Verschlüsselung im Klartext vor. Hinzu kommt die Tatsache, dass der Schlüssel dem Diensteanbieter, je nach verwendetem Verfahren, bekannt ist. Weitere Angriffspunkte könnten Hintertüren, schwache Algorithmen oder Implementierungsfehler darstellen. Daher setzt diese Lösung eine Vertrauensbeziehung zwischen Nutzer und Diensteanbieter voraus. Die Nutzer müssen dem Dropbox-Dienst vertrauen, dass dieser die Daten:

- sicher behandelt
- nicht mitliest
- nicht an Dritte weitergibt
- mit sicheren Algorithmen und Implementierungen schützt
- mit Hintertür-freien Lösungen verarbeitet

Diese Voraussetzungen schließen Dropbox und ähnliche Dienste von einer Firmennutzung aus, da in den meisten Geschäftsszenarien eine vertrauliche Datenbehandlung höchste Priorität hat.

An dieser Stelle setzt das Kozept unseres Papiers an, das eine sichere und effiziente Lösung für die verteilte Datenspeicherung vorschlägt und die Vertraulichkeit der Daten ortsunabhängig und auf beliebigen Plattformen gewährleistet. Dabei werden die Daten in verschlüsselter Form gespeichert und übertragen. Die Verschlüsselung und Entschlüsselung der Daten findet ausschließlich auf den Endgeräten der Nutzer statt - daher liegen die Daten nur beim Nutzer in entschlüsselter Form vor. Durch diese Maßnahme sind die Daten geschützt, selbst wenn unsichere Speichermedien (Cloud Storage) oder Übertragungsmedien (offene WLANs) genutzt werden. Der Diensteanbieter wird hierbei ausschließlich für das verlässliche Bereitstellen von Speicher benötigt, eine Vertrauensbeziehung ist nicht länger erforderlich.

In diesem Papier wird zuerst im Kapitel 2 eine Motivation für unseren Ansatz anhand von verschiedenen potenziellen Anwendungsszenarien gegeben. In Kapitel 3 wird ein Überblick über die grundlegenden Technologien gegeben. Kapitel 4 stellt das Konzept des Sec^2-Projektes vor. Abschließend gibt Kapitel 5 eine Zusammenfassung des vorgestellten Themas.

2 Potentielle Anwendungsszenarien

Grundsätzlich zielt die Architektur darauf ab, alle gängigen, kollaborativen Office-Prozesse in einer immer agiler werdenden Arbeitswelt abbilden zu können. Hierbei visiert die vorgestellte Lösung alle Anwendungsszenarien an, die durch eine oder mehrere der folgenden Anforderungen beschrieben werden können:

- Sichere Kommunikation im Sinne einer Online Präsentation (ähnlich WebEx oder Live Meeting)
- Sicherer Datenraum für eine spezifizierte Anwendergruppe (ähnlich Sharepoint)
- Absicherung der Daten auf Dokumentenebene und Einschränkung der Verwendbarkeit (Einschränkung der Benutzer, die auf die Daten zugreifen können)
- Daten sollen nicht auf unsicheren Plattformen gespeichert werden (Problem bei Cloud Storage)
- Ad-hoc Zugriff von mobilen Endgeräten möglich

Die genannten Punkte finden sich in vielen Anwendungsfällen wieder. Beispiele hierfür sind folgende:

- Vertrauliche Zusammenarbeit auf der Management-Ebene (z.B. Merger and Acquisition, Behörden)
- Kommunikation und Datenaustausch zwischen Herstellern und Zulieferern im Industriebereich
- Kommunikation und Datenaustausch auf Verbandsebene ohne zentrale Infrastruktur
- Privater Austausch von persönlichen Daten im Rahmen von sozialen Netzwerken (z.B. Fotos, die nur für eine bestimmte Gruppe sichtbar sind)

Die vorgestellte Architektur ermöglicht die Entwicklung von sicheren, mobilen Plattformen für unternehmensübergreifende Kollaboration und dezentrale Datenspeicherung. Gerade in diesem Umfeld werden in der Zukunft neue Lösungen gefragt sein, um den Anforderungen an eine nutzerfokussierte Datenspeicherung gerecht zu werden. Betrachtet man die Diskussion um die Sicherheit bisheriger Ansätze, so wird das Marktpotential der zu entwickelnden Lösung deutlich. Die gleiche Situation ist in den Bereichen der Kollaborationsplattformen (z.B. MS Sharepoint) und mobiler Anwendungen zu beobachten. Hierbei wird die rasante Marktentwicklung stets von der Frage nach der Sicherheit begleitet.

3 Verwendete Technologien

Die sich in der Entwicklung befindende Demonstrationsimplementierung basiert auf weit verbreiteten, offenen Standards. Als Basis für die Datenstrukturierung dient die *eXtensible Markup Language* [BPSMM+06]. Dieses plattform- und implementierungsunabhängige Format bietet eine gute Grundlage zur Bearbeitung und Abbildung strukturierter Daten und ist deswegen auch für unser Konzept gut geeignet.

XML Dokumente verfügen oftmals über vertrauliche Daten, die authentisch gehalten werden müssen. Um Vertraulichkeit, Authentizität und Integrität der XML Daten gewährleisten zu können wurden vom W3C Konsortium die Standards *XML Encryption* und *XML Signature* eingeführt. Diese Standards bieten Nachrichten-basierte Sicherheit. Im Vergleich zu PGP [ETLR01] oder S/MIME [Rams04] können hiermit auch beliebige Datenteile oder -fragmente geschützt werden.

3.1 XML Signature

Die XML Signature Spezifikation [ERSH+08] umschreibt Maßnahmen zur Gewährleistung von Authentizität und Integrität bezogen auf XML Daten. Ein Beispiel einer Nachricht, deren Daten mit einer XML Signature geschützt sind gibt Abbildung 1.

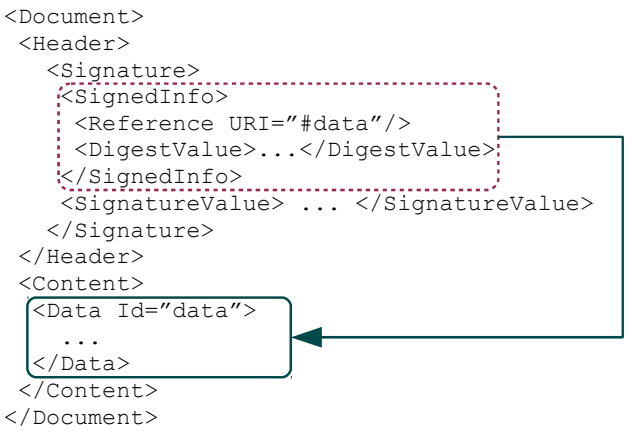

Abb. 1: Beispiel einer Nachricht mit einer XML Signature

Die dargestellte XML Nachricht enthält ein Element `<Data>`, das mittels einer XML Signature gesichert ist.
Die XML Signature ist innerhalb des Dokumenten `<Header>` Elements definiert und besteht aus zwei obligatorischen Elementen: `<SignedInfo>` und `<SignatureValue>`. Das `<SignedInfo>` Element enthält eine ID-Referenz, die auf das `<Data>`-Element verweist über das ein Hash-Wert berechnet wurde. Um das `<SignedInfo>`-Element zu schützen, wurde ein Signatur-Wert über dieses Element berechnet und dessen Wert als `<SignatureValue>` gesetzt. Dies wird typischerweise mit einem Public-Key-Verfahren, wie beispielsweise RSA oder DSA, durchgeführt.

3.2 XML Encryption

In der XML Encryption Spezifikation [ERID+02] werden Vorgehensweisen zum Schutz der Vertraulichkeit der XML Inhalte festgelegt. In den meisten Fällen wird dabei eine hybride Verschlüsselung eingesetzt. Dies bedeutet, dass der Sender die Nachricht in zwei Schritten bearbeitet:

1. Er wählt einen *Sitzungsschlüssel* k. Diesen Schlüssel verschlüsselt er z.B. mit einem öffentlichen Schlüssel (eines asymmetrischen Verfahrens) des Empfängers.
2. Er verschlüsselt die Daten mit dem *Sitzungsschlüssel* k unter Verwendung eines schnellen, symmetrischen Algorithmus.

Abbildung 2 gibt ein Beispiel einer XML Nachricht, die verschlüsselte Daten enthält. Sie besteht aus zwei Teilen:

1. `<EncryptedKey>`: Dieses Element enthält den `CipherValue`, welches den verschlüsselten Sitzungsschlüssel k darstellt.
2. `<EncryptedData>`. Dieses Element enthält die mit dem Sitzungsschlüssel verschlüsselten Daten. Der dafür verwendete Algorithmus ist im `<EncryptionMethod>`-Element definiert.

Der Algorithmus für die Sitzungsschlüsselverschlüsselung kann frei gewählt werden. Hierbei stehen wahlweise asymmetrische oder symmetrische Kryptosysteme zur Verfügung.

Innerhalb eines Dokuments können mehrere `<EncryptedKey>` Elemente definiert werden. Der Sitzungsschlüssel eines `<EncryptedKey>`-Elements kann für mehrere `<EncryptedData>` Elemente verwendet werden.

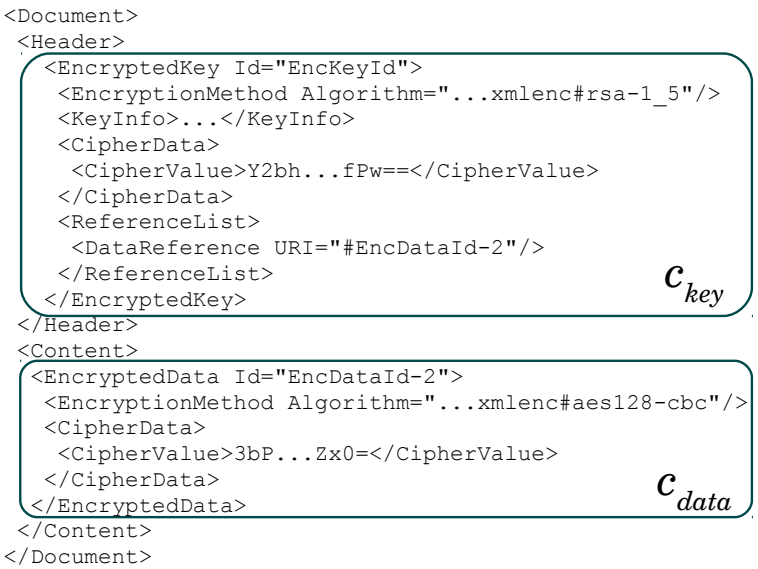

Abb. 2: Beispiel einer Nachricht mit verschlüsselten Daten

3.3 SAML

Da das Konzept ebenfalls Aspekte im Zusammenhang mit Single Sign-on Verfahren verwendet – um unnötige Mehrfachauthentifizierungen zu vermeiden – wurde die *Security Assertion Markup Language (SAML)* [OASI11] integriert. SAML ist ein auf XML beruhendes Framework, das es ermöglicht Sicherheitsaussagen direkt in XML zu modellieren. Es erlaubt das Abbilden ganzer Prozesse, wie z.B. Anmeldevorgänge oder die sichere Attestierung von Entitäts-Attributen durch eine vertrauenswürdige dritte Instanz. Darüber hinausgehend ist es möglich, die gesamte Kommunikation an vorgeschriebene Transportmedien zu binden.

Die Authentizität und Integrität der transportierten SAML-Tokens wird üblicherweise durch XML Signaturen gesichert.

4 Konzept

Die im Sec^2-Projekt zu erforschenden, innovativen Technologien für die mobile Sicherheit zeigen einen alternativen Weg zu einer sicheren Datenspeicherung. Ziel ist es, dass Anwender jederzeit und an jedem Ort sicher miteinander kommunizieren können. Dieses Ziel wird mit der Entwicklung eines sicheren Datenraumes für mobile Geräte durch lückenlose Verschlüsselung schützenswerter Daten erreicht. Die Daten können nach der Bearbeitung immer in *verschlüsselter Form* bei beliebigen Cloud Storage Anbieter abgelegt werden, wobei der mobile Client keine Vertrauensbeziehung zum Cloud-Server benötigt.

Nachfolgend wird eine Übersicht der Systemarchitektur gegeben und alle relevanten Komponenten erläutert. Anschliessend wird auf das Schlüsselkonzept und -management im Projekt eingegangen.

4.1 Systemarchitektur

Abbildung 3 zeigt einen Überblick der Systemarchitektur. Die Komponenten werden im Folgenden näher beschrieben:

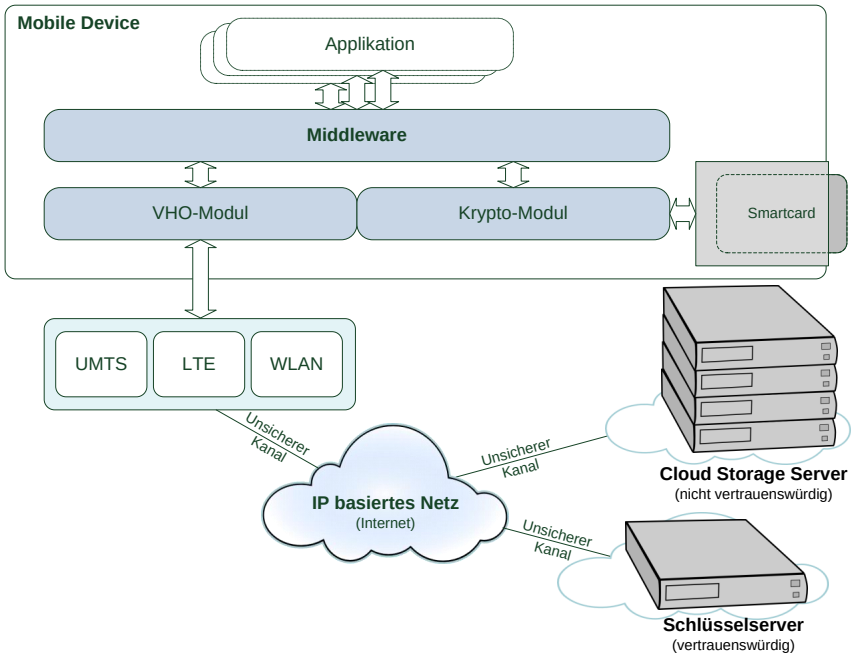

Abb. 3: Systemarchitektur

- **Middleware**. Die Middleware ist sowohl für die Verschlüsselung der ausgehenden als auch für die Entschlüsselung der eingehenden Nachrichten zuständig und bindet hierzu unterstützend eine Smartcard ein. Die Nutzung der Middleware ist für die gesicherten Applikationen transparent.
- **VHO-Modul**. Das Vertical Handoff (VHO) Modul stellt eine zuverlässige Verbindung der Geräte – auch beim Wechsel zwischen unterschiedlichen Übertragungsstandards – sicher. Somit wird dem mobilen Nutzer eine nahtlose, Cloud-basierte Dienstenutzung in einem heterogenen Netz (z.B. UMTS, LTE, WLAN) ermöglicht. Das Mobilitätsmodul verfolgt hierbei einen client-basierten Ansatz, bei der sich die komplette vertikale Mobilitätslogik auf mobiler Clientseite befindet. Dieser Ansatz hat den Vorteil, dass keine Änderungen seitens der Netzinfrastruktur und -betreiber vorgenommen werden müssen, was eine Kostenersparnis darstellt.
- **Krypto-Modul**. Das Kryptomodul stellt der Middleware den Zugriff auf die implementierten kryptographischen Primitive zur Verfügung. Zum einen werden kryptographische Mechanismen auf der mobilen Plattform selbst implementiert, zum Anderen wird eine programmierbare Smartcard eingesetzt.
- **Smartcard**. Da die Endgeräte – statt wie im herkömmlichen Modell die zentrale Server-Infrastruktur – in unserem Projekt Daten sowohl ver- als auch entschlüsseln, benötigen sie einen sicheren Speicher für kryptographische Schlüssel. In diesem Projekt wird die Hardware des Endgerätes (d.h. Smartphone, Tablet-PC) als *nicht vertrauenswürdig* betrachtet, daher kommt das permanente Ablegen der Schlüssel im Gerät selbst nicht in Frage. Da die meisten Endgeräte aber über micro-SD Kartenslots verfügen und bereits Smartcards in diesem Formfaktor existieren, bietet es sich an, auf diese Smartcards als Schlüsselspeicher zu setzen.
- **Speicherserver**. Die in der Cloud befindlichen Speicherserver stellen Speicherplatz für die verschlüsselten Nutzdaten bereit.

- **Schlüsselserver**. Für die Ent- und Verschlüsselung der bearbeiteten Daten muss der korrespondierende Schlüssel einmalig vom sicheren und vertrauenswürdigen Schlüsselserver bezogen werden. Alle Datenschlüssel werden dort – ebenfalls verschlüsselt – abgelegt und können nur auf den Endgeräten entschlüsselt werden. Näheres zur Schlüsselhierarchie findet sich in Abschnitt 4.2. Außerdem werden hier Autorisierungstabellen verwaltet, welche die Rechte der Anwender beschreiben und deren Zugehörigkeit zu verschiedenen Gruppen festlegen.

4.2 Schlüsselkonzept

Die Anforderungen an Gruppenkommunikation, Langzeitsicherheit und minimalem Datenaufkommen erfordern weitreichende Überlegungen hinsichtlich der relevanten Aspekte und eventueller, unerwünschter Seiteneffekte der eingesetzten Schlüsselhierarchie. Aus diesem Grund führen wir ein, im Cloud Computing bisher einzigartiges, Schlüsselkonzept ein, das eine unterschiedliche Kritikalität für einzelne Schlüssel und Schlüsseltypen im System vorsieht.

Zur Umsetzung basiert das vorgeschlagene Konzept auf einer Hierarchie aus drei Schlüsseltypen:

- **Benutzerschlüssel**. Jeder Benutzer ist im Besitz eines asymmetrischen Schlüsselpaares, welches für die sichere Übertragung von Gruppenschlüsseln und zur Autorisierung am Schlüsselserver verwendet wird.
- **Datenschlüssel**. Jeder Datensatz wird mit einem eigenen, symmetrischen Schlüssel verschlüsselt. Datenschlüssel werden in verschlüsselter Form in den Dokumenten abgelegt.
- **Gruppenschlüssel**. Datenschlüssel einer Menge von Datensätzen werden mit einem weiteren symmetrischen Schlüssel verschlüsselt. Gruppenteilnehmer können den Gruppenschlüssel vom Schlüsselserver beziehen.

Tabelle 1 verdeutlicht die Kritikalitätsabstufungen der einzelnen Schlüsseltypen.

Tab. 1: Schlüsselkritikalität

Schlüsseltyp	Kritikalität
Benutzerschlüssel	mittel
Datenschlüssel	hoch
Gruppenschlüssel	sehr hoch

Die Schlüsseltypen sind eng miteinander verzahnt und kommen initial alle in Kombination zur Verwendung. Im weiteren Verlauf einer Gruppensitzung ist es nicht nötig alle Schlüssel zu verwenden, statt dessen kann ein verkürztes Verfahren benutzt werden um den Arbeitsablauf effizienter und schneller zu gestalten. Im Folgenden wird die Interaktion der unterschiedlichen Schlüsseltypen näher beleuchtet und die erforderlichen Schritte zur erfolgreichen Entschlüsselung eines Dokuments skizziert.

4.2.1 Prinziperläuterung anhand eines XML Dokuments

Abbildung 4 zeigt die Struktur der verwendeten XML Dokumente. Jedes Dokument enthält einen Block mit unverschlüsselten Meta Daten. Die Meta Daten beschreiben den Dokumenteninhalt und ermöglichen eine Suche und Klassifizierung anhand von Stichworten oder Beschreibungstexten. Die Struktur der Meta Daten ist frei wählbar und somit für alle Anwendungsfälle individualisierbar.

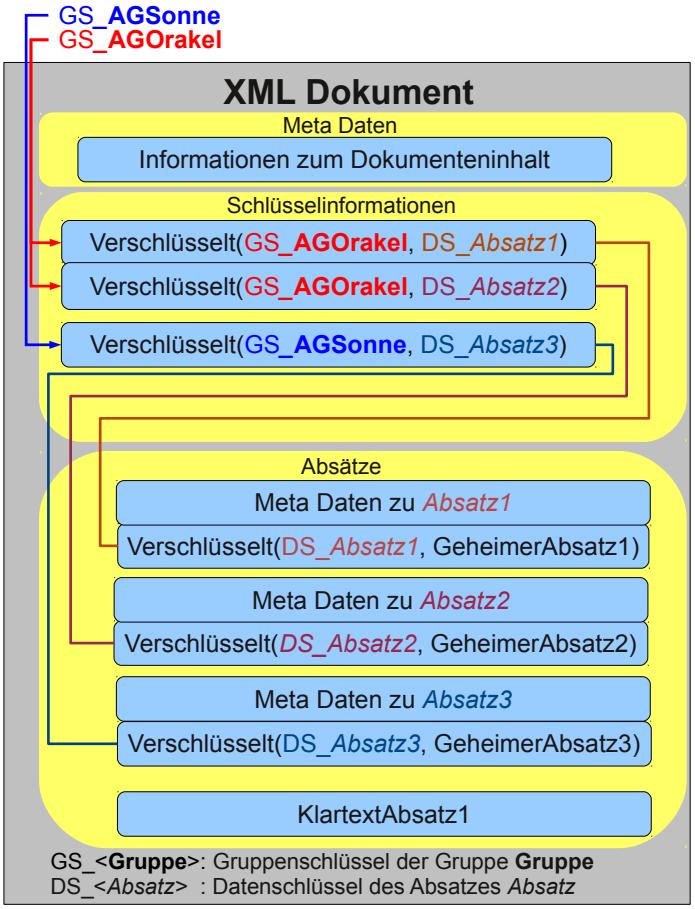

Abb. 4: Schematische Darstellung der verwendeten XML Dokumente

Nachfolgend beginnt der Schlüsselblock. In diesem sind die verwendeten Datenschlüssel, mit denen die einzelnen Nutzdatenblöcke gesichert sind, in verschlüsselter Form abgelegt. Die Entschlüsselung der Datenschlüssel ist nur unter Kenntnis des verwendeten Gruppenschlüssels möglich. Mit Hilfe der Datenschlüssel können die korrespondierenden Nutzdatenblöcke entschlüsselt werden.

Als letzter Gesamtblock im Dokument finden sich die einzelnen Nutzdatenblöcke. Sie enthalten, optional in unverschlüsselter Form, den eigentlichen Inhalt des Dokuments. Es steht dem Anwender frei Nutzdatenblöcke zu verschlüsseln oder auch vereinzelte Blöcke unverschlüsselt abzulegen. Durch die optionale unverschlüsselte Ablage wird eine maximal mögliche Flexibilität garantiert und die Verwendbarkeit der Dokumente in Applikationen, die nicht auf die Sec^2 Architektur zurückgreifen, sichergestellt. Das System kann folglich auch absolut transparent verwendet werden, z.B. in Anwendungsszenarien wo die XML Dokumente an Applikationen weitergereicht werden die keinen Gebrauch von den Verschlüsselungsfunktionen machen. Somit ist auch die Pflege von Status-,Verwaltungs- und Dokumenteninformationen über die zuvor eingeführten Meta Daten hinaus möglich.

4.2.2 Interaktion der Schlüsseltypen

Das Ablaufdiagramm in Abbildung 5 zeigt den Ablauf einer Entschlüsselung aus Sicht eines Benutzers und verdeutlicht die notwendigen Schritte und die Interaktion der verschiedenen Schlüsseltypen.

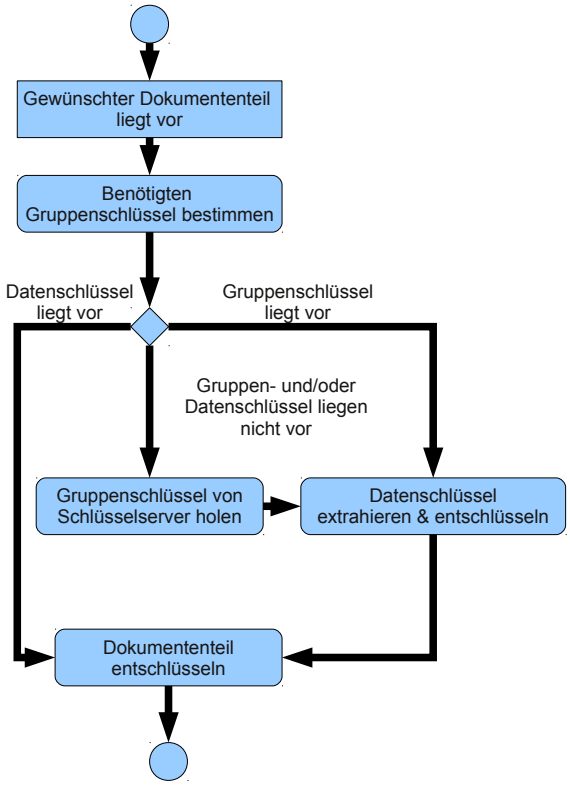

Abb. 5: Schlüsselprozedur

Um ein erhaltenes Dokument entschlüsseln zu können muss das Endgerät im Besitz eines oder mehrerer, für das jeweilige Dokument notwendigen, Gruppenschlüssel zur Entschlüsselung der Datenschlüssel sein.

Der Prozess der Entschlüsselung beginnt mit der Bestimmung der benötigten Datenschlüssel und die zur Entschlüsselung notwendigen Gruppenschlüssel. Hierzu wird zunächst der lokale Schlüsselspeicher durchsucht. Finden sich hier bereits referenzierte Datenschlüssel, so wird deren Beschaffung übergangen. Für fehlende Datenschlüssel wird wie im Folgenden beschrieben verfahren.

Die Schlüsselreferenzen des Datenschlüsselblocks werden untersucht. Hierzu erstellt die Client Anwendung eine Liste der benötigten Gruppenschlüssel. Der lokale Schlüsselspeicher wird auf die Verfügbarkeit der in der Liste aufgeführten Gruppenschlüssel durchsucht. Befinden sich notwendige Gruppenschlüssel bereits im lokalen Schlüsselspeicher so werden diese von der Beschaffung der Schlüssel vom Schlüsselserver ausgeschlossen.

Für fehlende Gruppenschlüssel erfolgt eine Anfrage beim Schlüsselserver. Hierzu wird eine, mit dem Benutzerschlüssel, signierte SAML Anfrage gesendet, die die Übertragung der fehldenen Schlüssel beantragt. Anhand der Signatur ist eine eindeutige Authentifizierung am Schlüsselserver möglich. Schutz vor Replay-Attacken bieten in die Signatur miteinbezogene Zeitstempel einer jeden Anfrage.

Der Schlüsselserver prüft jede Anfrage auf unterschiedliche Kriterien:

- Validität der Signatur
- Berechtigungen des Anfragenstellers
- Gruppenzugehörigkeit zum angefragten Gruppenschlüssel

Gruppenschlüssel werden ausschließlich an den Anfragensteller ausgegeben sofern alle genannten Kriterien ausreichend erfüllt sind. Konkret bedeutet dies, dass der Antragsteller eine valide Signatur über die Anfrage erstellt haben muss, die Berechtigung zum Anfragen von Gruppenschlüsseln besitzt und Mitglied der Gruppe ist dessen Gruppenschlüssel er gerade anfragt.

Erfüllt die Anfrage die beschriebenen Kriterien so verschlüsselt der Schlüsselserver den/die angefragten Gruppenschlüssel mit dem öffentlichen Teil des Benutzerschlüssels des Anfragenstellers und sendet die verschlüsselten Daten als Teil der, ihrerseits mit dem privaten Schlüssel des Schlüsselservers signierten, SAML Antwort. Im Fehlerfall erhält der anfragende Client eine SAML Antwort mit Fehler-/Statusinformationen. Die auf diese Weise erhaltenen Gruppenschlüssel werden auf Empfängerseite entschlüsselt und im lokalen Schlüsselspeicher abgelegt.

Liegen alle benötigten Gruppenschlüssel vor kann mit der Entschlüsselung der Datenschlüssel begonnen werden, sofern diese nicht bereits im Schlüsselspeicher vorhanden sind oder der Gruppenschlüssel nicht verfügbar ist. Die erhaltenen Datenschlüssel werden ebenfalls lokal abgelegt.

Als letzten Schritt werden alle Dokumententeile für die der entsprechende Datenschlüssel beschafft werden konnte entschlüsselt und liegen hier nach der Client Applikation im Klartext vor. Es ist der Applikation freigestellt in welchem Umfang sie nicht entschlüsselbare Dokumententeile, weil zB. der Gruppenschlüssel nicht beschafft werden konnte, verarbeitet.

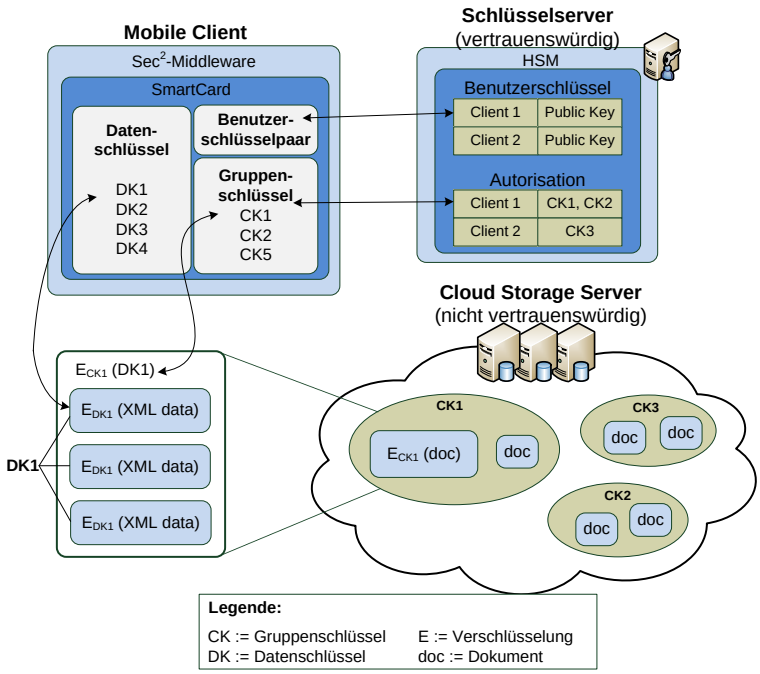

Abb. 6: Sec^2 Architekturübersicht

Die Verzahnung der einzelnen Schlüssel und deren Interaktion im Gesamtkonzept veranschaulicht noch einmal Abbildung 6.

5 Fazit

In diesem Papier haben wir das Konzept einer sicheren Datenablage beschrieben. Diese erlaubt es die Daten auf beliebigen Cloud-basierten Speicherdiensten innerhalb einer erstellten Gruppe zu speichern oder dynamisch auszutauschen. Da die Verschlüsselung innerhalb des Geräts automatisiert stattfindet bleibt die vollständige Kontrolle über die abgelegten Daten bei den Benutzern. Die verschlüsselten Daten können von keinen unautorisierten Benutzern entschlüsselt werden.

Im Rahmen des, vom Bundesministerium für Bildung und Forschung geförderten, Sec^2-Projekts wird das vorgestellte Konzept auf einer Android-Platform implementiert. Auf Seite des Schlüsselservers wird auf Hardware Security Module zurückgegriffen um eine maximal mögliche Sicherheit zu garantieren.

Desweiteren wird ein Konzept für eine Realisierung des Sec^2-Projekts ohne Verwendung eines zentralen Schlüsselservers entwickelt, welches den Benutzern eine dynamische Gruppenbildung und ein dynamisches Schlüsselmanagement erlaubt.

Literatur

[BPSMM+06] Bray, Paoli, Sperberg-McQueen, Maler, Yergeau, Cowan: Extensible Markup Language (XML) 1.1 (Second Edition). http://www.w3.org/TR/2006/REC-xml11-20060816/ (2006).

[Brod08] Brodkin: Gartner: Seven cloud-computing security risks. http://folk.ntnu.no/oztarman/tdt60/cloud computing/3 Cloud_Computing_Security_Risk.pdf (2008).

[Bök11] Böken: Fallstricke beim Cloud Computing. http://www.handelsblatt.com/technologie/it-tk/special-cloud-computing/fallstricke-beim-cloud-computing/4031642.html (2011).

[Drop11] Dropbox: Dropbox - Simplify your life. http://www.dropbox.com (2011).

[ERID+02] Eastlake, Reagle, Imamura, Dillaway, Simon: XML Encryption Syntax and Processing. http://www.w3.org/TR/xmlenc-core (2002).

[ERSH+08] Eastlake, Reagle, Solo, Hirsch, Roessler, Bartel, Boyer, Fox, LaMacchia, Simon: XML Signature Syntax and Processing (Second Edition). http://www.w3.org/TR/xmldsig-core (2008).

[ETLR01] M. Elkins, D. D. Torto, R. Levien, T. Roessler: MIME Security with OpenPGP. RFC 3156 (Proposed Standard) (2001), .

[LLC11] A. W. S. LLC: Amazon Simple Storage Service (Amazon S3). http://aws.amazon.com/s3 (2011).

[OASI11] OASIS: SAML XML.org — Online community for the Security Assertion Markup Language (SAML) OASIS Standard. http://saml.xml.org (2011).

[Rams04] B. Ramsdell: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. RFC 3851 (Proposed Standard) (2004), , obsoleted by RFC 5751.