Sicherer Schlüssel- und Informationsaustausch mit SAML

Dennis Felsch · Thorsten Schreiber
Christopher Meyer · Florian Feldmann · Jörg Schwenk

Horst Görtz Institut für IT-Sicherheit
Ruhr-Universität Bochum
{dennis.felsch | thorsten.schreiber | christopher.meyer
florian.feldmann | joerg.schwenk}@ruhr-uni-bochum.de

Zusammenfassung

Um Daten sicher auf cloud-basierten Speichersystemen ablegen zu können, müssen verschiedene Schutzmaßnahmen ergriffen werden. Eine dieser Schutzmaßnahmen besteht in der Verschlüsselung der Daten. Damit eine derartige auf Verschlüsselung beruhende Sicherheitsarchitektur gemeinschaftlich genutzt werden und anwenderfreundlich funktionieren kann, wird ein System zur effektiven Schlüsselverwaltung benötigt. Die vorliegende Arbeit befasst sich mit der Schlüsselverteilung, die ein Kernproblem der Schlüsselverwaltung darstellt. Um kryptografische Schlüssel sowohl authentifiziert als auch vertraulich übertragen zu können, wird eine Erweiterung zum etablierten SAML (Security Assertion Markup Language) Standard vorgeschlagen. Nach Entwicklung und Analyse wird die Anwendbarkeit dieser Erweiterung anhand einer Fallstudie gezeigt. Das vorgestellte Konzept ist vollständig zum SAML Standard kompatibel und ermöglicht daher die Benutzung bereits vorhandener Bibliotheken.

1 Einführung

Das SAML-Framework (Security Assertion Markup Language [RHPM+08]) bildet einen Grundpfeiler vieler aktueller Single Sign-On Systeme, da es einen flexiblen Mechanismus darstellt, um Authentifizierungsinformationen an Identitäten zu binden. Um die Sicherheit auch Inhaltsebene zu gewährleisten nutzt SAML die XML Signature- und Encryption-Standards ([ERSH+13], [ERHR13]). SAML kann jedoch durch seine Anpassungsfähigkeit weitaus mehr als nur den Austausch von Identitätsinformationen leisten. Die vorliegende Arbeit präsentiert einen Entwurf, der es erlaubt, mittels SAML einen authentifizierten Austausch kryptografischer Schlüssel umzusetzen. Dabei ist sichergestellt, dass alle Erweiterungen vollständig kompatibel zum SAML Standard sind, was es erlaubt, bestehende Implementierungen ohne Modifikationen weiter zu verwenden. Auf kanalsichernde Verfahren, wie beispielsweise SSL/TLS, kann vollständig verzichtet werden.

1.1 Bisherige Arbeiten

Techniken für die Bindung und Verteilung von Schlüsseln finden bereits große Verwendung: Mit Kerberos [NYHR05], dem de facto Standard für das Management und die Verteilung von *symmetrischem* Schlüsselmaterial, existiert bereits eine etablierte Lösung zum Schlüsselaustausch in verteilten Netzwerken. Systeme wie das weit verbreitete Microsoft Active Directory [MiTe]

P. Schartner · P. Trommler (Hrsg.) · D•A•CH Security 2013 · syssec (2013) 248-259.

basieren auf dem Kerberos Konzept. Aufgrund der Popularität von Kerberos enthält der SAML Standard Mechanismen zur Adaptierung des Kerberos-Konzepts in der Web-Browser Welt [HKHS10]. Ein großer Nachteil dieser Vorgehensweise ergibt sich aus der Fokussierung auf die Verteilung von ausschließlich symmetrischem Schlüsselmaterial.

Auch Systeme, die Maßnahmen zur Verteilung von asymmetrischem Schlüsselmaterial beinhalten, existieren bereits seit Jahren (bspw. OpenPGP [CDFS+07]). Hiermit ist es im Gegensatz zu Kerberos auch möglich, Schlüsselmaterial fest an Identitäten zu binden - zum Beispiel durch den Einsatz digitaler Signaturen. Diese Vorgehensweise ist analog zu der des X.509-Standards [CSFB+08], der die Basis für SSL/TLS gesicherte Kommunikation bildet. Ein ähnliches Ziel verfolgt das XKMS-Protokoll [HaMy05], indem das Konzept von OpenPGP auf die Web-Service Technologie übertragen wird. Hierdurch wird eine Art Abrufservice, wie bereits von den Schlüsselservern aus OpenPGP bekannt, realisiert.

Allen bisherigen Lösungen gemein ist, dass sie sich auf einen bestimmten Typ Schlüsselmaterial festlegen, was die Flexibilität der Systeme massiv einschränkt. Weiterhin bietet allein das SAML-Kerberos-Profil eine Lösung an, die auch im Umfeld von Browsern nutzbar ist. Gerade im Hinblick auf die Einführung von WebCrypto API [web] und ähnlichen Technologien ist eine universelle Lösung notwendig, die eine nutzerbezogene Anwendung von Kryptografie ermöglicht und unabhängig von Schlüsseltypen und Client-Software einsetzbar ist. Aufgrund der steigenden Verbreitung von Social Networking und Social Media und den daraus resultierenden Anforderungen an den Datenschutz und die nutzerfokussierte Datenkontrolle erwarten die Autoren einen steigenden Bedarf für diesbezügliche Lösungsansätze. Um einen Bezug zwischen Nutzer und Schlüsselmaterial unzweifelhaft herstellen zu können, ist es unabdingbar, dem Aspekt der Schlüssel-zu-Identitäten Bindung besondere Aufmerksamkeit zu schenken. Darüber hinaus darf die Flexibilität nur in einem für die Sicherheit notwendigen Maße eingeschränkt werden, um die Lösung universell anwendbar und verfügbar zu machen.

1.2 Innovation

Die hier vorgeschlagene Lösung ermöglicht einen auf dem SAML Framework basierenden standardisierten, flexiblen und geschützten Schlüsselaustausch. Hierzu wird eine Erweiterung vorgeschlagen, die keinerlei Modifikationen am SAML Standard erfordert, sondern sich in SAML-spezifische Erweiterungspunkte einfügt. Dies erlaubt eine nahtlose Integration, ohne bestehende XML-Schemata oder Standards zu verletzen. Durch die hier beschriebene Erweiterung für den SAML Standard kann sowohl symmetrisches als auch asymmetrisches Schlüsselmaterial an SAML Assertions gebunden, kryptografisch gesichert und integritätsgeschützt transportiert werden. Somit wird gezielt einer Fragmentierung durch eine Vielzahl an herstellerspezifischen und untereinander inkompatiblen Eigenlösungen beim Schlüsselaustausch vorgebeugt. Weiterhin kann während der Schlüsselaustauschphase explizit auf einen Transportebenenschutz in Form von Verschlüsselung (z.B. durch die Verwendung von SSL/TLS) verzichtet werden.

2 Technische Grundlagen

Die vorliegende Arbeit basiert auf offenen Standards, die in diesem Abschnitt erläutert werden.

2.1 Datentransport

Alle Nachrichten werden über eine Transportschicht auf Ebene 4 des OSI-Referenzmodells [Tane03] übermittelt. Diese Ebene erfüllt keine Sicherheitsziele wie Vertraulichkeit oder In-

tegrität, d.h. sie ist ein unsicherer Kanal und somit auslesbar und manipulierbar. Eine Nachrichtenebene, die auf diese Schicht aufgesetzt wird, muss folglich Massnahmen für benötigte Sicherheitsziele selbst implementieren.

Für die Kommunikation über diese Transportschicht wird das Protokoll HTTP [FGMF+99] angewendet. In diesem textbasierten Protokoll wird das Format einer Anfrage eines Clients und ihrer Antwort vom Server beschrieben. Anfragen über HTTP sind ohne weitere Hilfsmittel zustandslos und müssen so alle Informationen enthalten, die für die Antwort notwendig sind.

2.2 Zertifikate

Zertifikate sind signierte Datenstrukturen, in denen Identitäten kryptografisch sicher an einen Schlüssel gebunden werden. Der X.509 Standard [CSFB+08] beschreibt eine solche Datenstruktur sowie eine zum zuverlässigen Betrieb notwendige Infrastruktur. Diese Public Key Infrastruktur (PKI) ist notwendig für die Ausstellung, Verteilung und Validierung von Zertifikaten.

2.3 Sicherheitsmechanismen in XML

Mit XML Signature [ERSH+13] wurde durch eine Arbeitsgruppe des *World Wide Web Consortium* ein Standard geschaffen, der eine dem PKCS#7 Standard [JeLS09] ähnliche digitale Signatur in XML-Dokumenten definiert. Für die Verifikation von Signaturen enthält XML Signature bereits alle notwendigen Datenstrukturen für die Verteilung, das Management und den Transport von Schlüsseln. Ein weiterer Standard dieses Gremiums ist XML Encryption [ERHR13], mit dem XML-Dokumente verschlüsselt werden können. Da ein Chiffrat hierbei in valides XML eingebettet wird, ist es möglich, einzelne Dokumententeile zu verschlüsseln oder mehrere Chiffrate in einem XML-Dokument zusammenzufassen. Beide Standards sind eng miteinander verknüpft und profitieren so von den Datenstrukturen des jeweils anderen.

Um bekannt gewordenen Angriffen auf XML Signature und XML Encryption (vgl. [JaSo11, JaPS13]) vorzubeugen, empfiehlt dieser Erweiterungsvorschlag explizit die Verwendung der in Version 1.1 der Standards eingeführten GCM-Betriebsmodi (Galois Counter Mode) sowie den Verzicht auf RSA-PKCS#1 v1.5 als Verfahren zur Schlüssel-Verschlüsselung.

2.4 SAML

Die *Security Assertion Markup Language (SAML)* ist eine Spezifikation der *Organization for the Advancement of Structured Information Standards (OASIS)*. SAML definiert die Syntax, die Verarbeitung und den Austausch von sicherheitsrelevanten Aussagen (*Assertions*) von Systeminstanzen über ein Subjekt [RHPM+08]. Dieser offene Standard basiert auf XML und ist durch die effiziente und sichere Umsetzung von Single Sign-On Szenarien weitreichend bekannt und akzeptiert. Die Definition von SAML ist jedoch absichtlich sehr abstrakt gehalten, wodurch der Standard potentiell wesentlich breitere Anwendung finden kann.

Die Bestandteile von SAML sind hierarchisch gegliedert. *Profiles* stellen die höchste Ebene dar, darunter liegen *Bindings*, *Protocols* und *Assertions*, wie in Abbildung 1 gezeigt. Ein Profile im Kontext von SAML ist eine bestimmte Vorschrift, wie Assertions, Protocols und Bindings zu kombinieren sind, um einen vordefinierten Anwendungsfall zu erfüllen. Das in der vorliegenden Arbeit benutzte *Assertion Query/Request Profile* dient zum Abfragen von Informationen, über die eine Systemautorität verfügt. Das Profile definiert eine Anfrage (*AttributeQuery*) und ihre Antwort (*Response*).

Ein Binding ist in SAML eine Definition, wie der Transport einer Anfrage und der dazugehörigen Antwort in ein bereits existierendes Standardprotokoll eingebunden werden kann. In der vorliegenden Arbeit wird das *HTTP-POST-Binding* benutzt, das eine Übertragung der Nachrichten mittels des HTTP-Protokolls vorschreibt.

Durch ein SAML Protocol wird beschrieben, welche Informationen in der Anfrage und Antwort enthalten sein müssen. Diese Arbeit nutzt das *Assertion Query/Request Protocol*. Dabei wird eine Anfrage gestellt, die eine gewünschte Ressource identifiziert. Diese Anfrage wird mit einer entsprechenden Assertion beantwortet, die eingebettet in ein Response Element die gewünschte Ressource enthält.

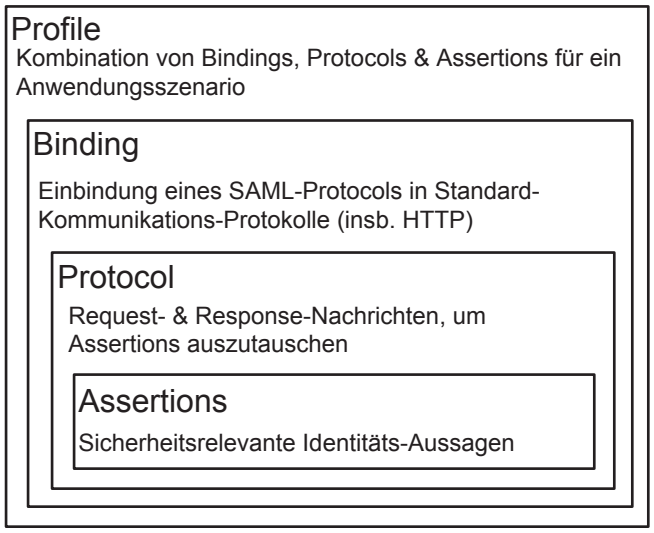

Abb. 1: Strukturebenen von SAML (nach [RHPM+08, S. 16])

Die sicherheitsrelevanten Informationen, die auszutauschen sind, werden durch eine Assertion ausgedrückt (vgl. Abbildung 2). In dieser wird eine Aussage über ein bestimmtes Subjekt von einer Autorität getätigt. Eine Assertion wird als XML-Teilbaum mit dem Wurzelelement `Assertion` abgebildet. In einer Assertion können drei Typen von Statements enthalten sein, die Vorlagen für die Abbildung von häufig auftretenden Informationstypen sind. Als Typen stehen *Authentication Statement, Attribute Statement* und *Authorization Decision Statements* zur Verfügung. Von diesen Möglichkeiten ist für die vorgeschlagene Lösung nur das AttributeStatement relevant. Dessen Kindelemente dürfen aus einer beliebigen Anzahl von `Attribute` Elementen bestehen, die durch das XML-Attribut `name="..."` identifiziert werden. Der zugehörige Wert wird in einem untergeordneten `AttributeValue` Element abgelegt. `AttributeValue` akzeptiert ein beliebiges XML-konformes Fragment als Kindelement, wodurch es zu einem wichtigen Erweiterungspunkt des SAML Standards wird.

Eine Assertion enthält die Elemente `Subject`, `Issuer` und `Conditions` mit Informationen, über welches Subjekt die Aussage getroffen wird, von welcher Instanz diese Aussage ausgeht und wann bzw. wie lange sie gültig ist. Danach folgt eine Anzahl von *Statements* mit den sicherheitsrelevanten Informationen. Diese Aufzählung nötiger Informationen wird kryptographisch durch die Verwendung der XML Security Standards abgesichert. Um das Sicherheitsziel Authentizität zu erreichen, wird der XML Signature Standard benutzt. Das entsprechende Feld `Signature` enthält die Signatur und bezieht sich als Enveloped Signature durch eine Refe-

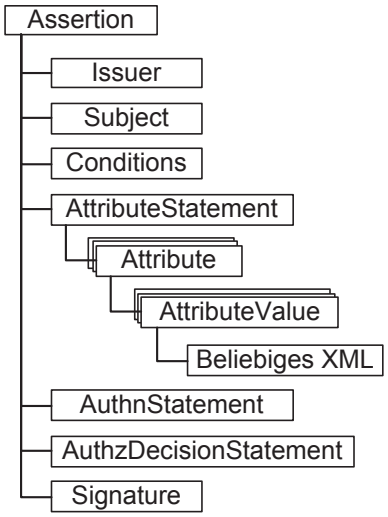

Abb. 2: Schematische Darstellung einer Assertion

renz auf das Elternelement `Assertion`. Ist die Assertion in eine Response eingebunden, ist es sinnvoll, die Signatur auf das `Response` Element auszudehnen und so die ganze Nachricht zu authentifizieren. Falls die in den Statements erwähnten Informationen vertraulich behandelt werden sollen, können die im `AttributeStatement`-Block enthaltenen `Attribute` Elemente mit XML Encryption verschlüsselt werden. Die nun verschlüsselten Elemente werden dann bis zur Entschlüsselung in `EncryptedAttribute` umbenannt.

3 Konzept

Das zentrale Ziel dieses Erweiterungsvorschlags für SAML besteht darin, das in SAML genutzte Konzept der Bindung von Authentifizierungs-Informationen an Identitäten auf allgemeine Informationen und insbesondere kryptografische Schlüssel zu erweitern. Im Folgenden wird detailliert auf die Kommunikation beim Austausch kryptografischer Schlüssel eingegangen. Das allgemeine Prinzip lässt sich jedoch einfach auf andere Inhalte übertragen.

3.1 Szenario

Folgende Ausgangskonstellation wird angenommen: Ein Client (Key Requestor, KR) benötigt einen Schlüssel, den ein Schlüsselserver (Key Server, KS) besitzt. Beide Kommunikationspartner besitzen jeweils ein asymmetrisches Schlüsselpaar und haben ihre öffentlichen Schlüssel bereits im Vorfeld ausgetauscht oder sind anderweitig in der Lage, die Vertrauenswürdigkeit des öffentlichen Schlüssels des jeweils anderen zu prüfen.

Um den Schlüssel anzufordern, sendet der KR eine SAML AttributeQuery an den KS. Über diese Anfrage erstellt der KR eine XML Signatur. Durch das `Issuer`-Element und die digitale Signatur ist die Anfrage nun fest an die Identität des KR gebunden. Die Anfrage des KR enthält eine Referenz auf den Schlüssel (z.B. Name, ID), den der KR abrufen möchte. Diese Referenz kann optional mithilfe von XML Encryption geschützt werden, falls sie eine vertrauliche Information darstellt.

Bei der Verarbeitung der Anfrage prüft der KS zuerst die Signatur der Anfrage. Ist sie valide, entscheidet der KS ob der KR über ausreichende Rechte verfügt, um den Schlüssel abzurufen.

Im positiven Fall erstellt der KS eine SAML Response mit einer eingebetteten Assertion, die den angeforderten Schlüssel in verschlüsselter Form enthält. Zur Verschlüsselung verwendet der KS den öffentlichen Schlüssel des KR, um nur ihm die Entschlüsselung zu ermöglichen. Der KS signiert die Assertion, bevor er sie an den KR zurückschickt. Somit ist auch die Antwort fest an die Identität von KS gebunden.

3.2 Technische Details

Um die Schlüssel zu übertragen, werden die in den XML Signature- und XML Encryption Standards spezifizierten Elemente genutzt, die ohnehin im SAML Standard Anwendung finden. Um hierbei die Flexibilität bezüglich des Schlüsseltyps zu garantieren, wird als Container Element `KeyInfo` aus dem XML Signature Standard gewählt. Dieses Element kann benutzt werden, um öffentliche Schlüssel verschiedener kryptografischer Verfahren zusammen mit Metadaten abzubilden. Für symmetrische Schlüssel wird das `EncryptedKey` Element aus dem XML Encryption Standard verwendet. Die Struktur dieses Elements besteht im Wesentlichen aus den Kindelementen `KeyInfo`, `EncryptionMethod`, `CarriedKeyName` und `CipherData`. An dieser Stelle beinhaltet `KeyInfo` die Information, welcher Schlüssel für die Verschlüsselung des auszutauschenden Schlüssels verwendet wurde. Durch `EncryptionMethod` wird der Algorithmus identifiziert, der zur Verschlüsselung eingesetzt wurde, während die Bezeichnung des symmetrischen Schlüssels in `CarriedKeyName` angegeben wird. `CipherData` enthält im `CipherValue` Element den verschlüsselten Schlüssel.

Für die Übertragung anderer Informationen können statt `KeyInfo` ebenfalls bereits standardisierte, auf XML basierende Datenstrukturen verwendet werden, sofern solche bereits existieren.

3.3 Kommunikationsablauf

Für die Anfrage des KR wird ein `AttributeQuery` Element verwendet, das mithilfe eines HTTP-POST-Requests an den KS gesendet wird. Ein Beispiel für eine solche Anfrage ist in Abbildung 3 und Abbildung 4 zu sehen. Die *Enveloped Signature* referenziert das Wurzel-Element und sichert so die gesamte Anfrage. Durch das in der signierten Anfrage enthaltene `Issuer` Element wird die Identität des KR an die Anfrage gebunden. Ebenso enthält diese mit dem Element `EncryptedAttribute` eine verschlüsselte Referenz (z.B. Name, ID) auf den gewünschten Schlüssel. Da ein `AttributeQuery` Element laut SAML-Definition kein `EncryptedAttribute` Element aufnehmen kann, muss diese Einschränkung mit einer Kapselung durch ein `Attribute` und ein `AttributeValue` Element umgangen werden.

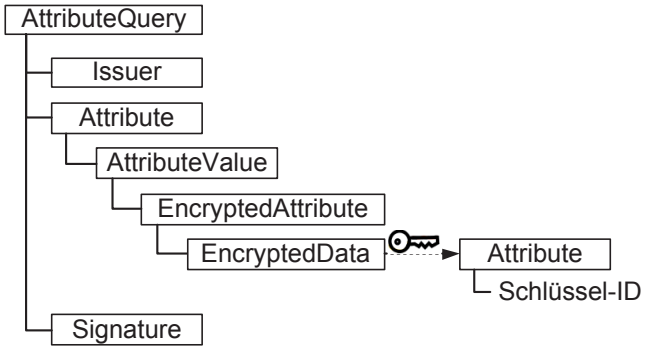

Abb. 3: Schematische Darstellung einer AttributeQuery

```
<?xml version="1.0" encoding="utf-8"?>
<saml2p:AttributeQuery xmlns:saml2p="...:SAML:2.0:protocol"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 xmlns:saml2="...:SAML:2.0:assertion"
 ID="query1" Version="2.0">
  <saml2:Issuer>[ID des KR]</saml2:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:SignatureMethod Algorithm=".../xmldsig-more#rsa-sha256" />
      ...
      <ds:Reference URI="#query1">
        ...
        <ds:DigestMethod Algorithm=".../xmlenc#sha256" />
        <ds:DigestValue>
          [SHA-256 Hash-Wert der Anfrage]
        </ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
      [Signatur-Wert]
    </ds:SignatureValue>
    <ds:KeyInfo>
      [Referenz auf Schlüsselpaar/Zertifikat des KR]
    </ds:KeyInfo>
  </ds:Signature>
  <saml2:Attribute Name="[zufällige ID]">
    <saml2:AttributeValue>
      <saml2:EncryptedAttribute>
        <xenc:EncryptedData Type=".../xmlenc#Element">
          <xenc:EncryptionMethod Algorithm=".../xmlenc11#aes128-gcm" />
          ...
          <xenc:CipherData>
            <xenc:CipherValue>
              <!-- verschlüsselt -->
              <saml2:Attribute Name="[Key-ID, Key-Name, o.ä.]" />
            </xenc:CipherValue>
          </xenc:CipherData>
        </xenc:EncryptedData>
      </saml2:EncryptedAttribute>
    </saml2:AttributeValue>
  </saml2:Attribute>
</saml2p:AttributeQuery>
```

Abb. 4: XML Darstellung einer AttributeQuery

Die Antwort des KS besteht aus einem `Response` Element, das eine `Assertion` enthält. Diese wiederum enthält ein `Attribute` Element, das den gewünschten Schlüssel nach dem oben beschriebenen Konzept beinhaltet. Dieses Element ist mit XML Encryption gesichert. In der Folge ist der gewünschte Schlüssel doppelt verschlüsselt, einmal im `EncryptedKey` und ein weiteres Mal im `EncryptedAttribute`. Dies ist notwendig, um nicht nur die Vertraulichkeit des eigentlichen Schlüssels sondern auch die seiner Metadaten zu gewährleisten. Durch das `Issuer` und das `Subject` Element der Assertion sowie der digitalen Signatur ist der zu übertragende Schlüssel nun fest an die Identitäten von KR und KS gebunden. Die schematische Darstellung der `Response` Nachricht findet sich in Abbildung 5, sowie die XML Darstellung in Abbildung 6.

4 Fallstudie

Die vorgestellte Anwendung und Erweiterung des SAML Frameworks ist wesentlicher Bestandteil der im Forschungsprojekt *Sec*^{2} entwickelten Architektur. *Sec*^{2} ist eine Kooperation

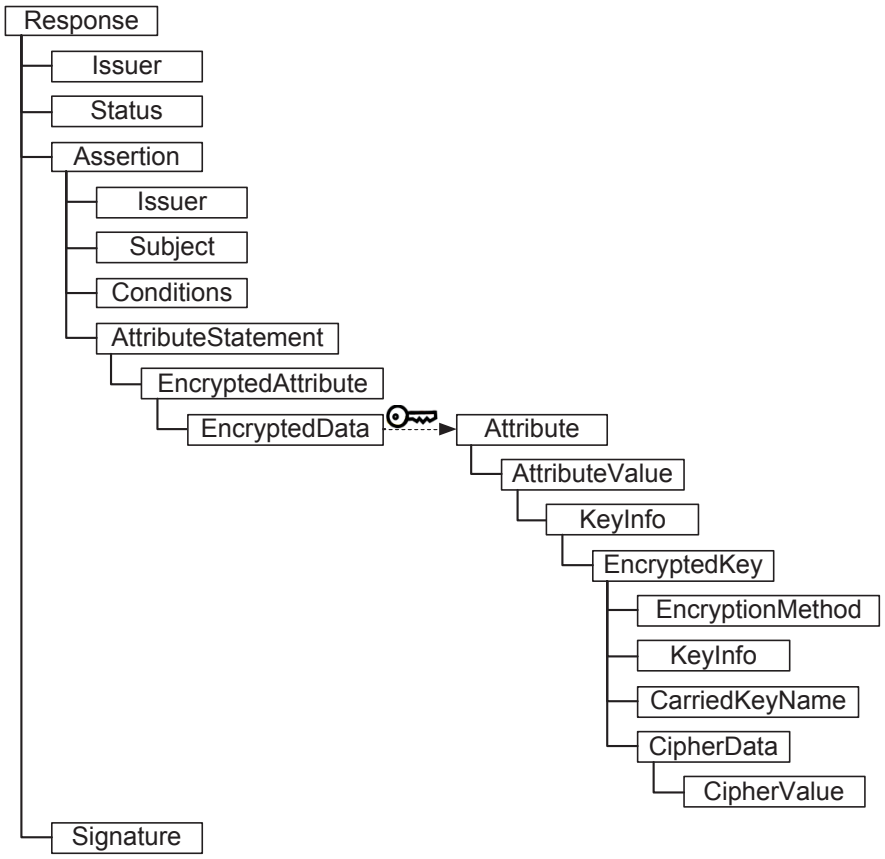

Abb. 5: Schematische Darstellung einer Response

aus Universität und Industrie, die vom Bundesministerium für Bildung und Forschung gefördert wird. Die Projektträgerschaft wurde initial durch das Deutsche Zentrum für Luft- und Raumfahrttechnik ausgeübt, nach einem Wechsel des Projektträgers unterliegt das Projekt nun der Betreuung durch die VDI/VDE Innovation + Technik GmbH, Berlin.

4.1 Übersicht der Architektur

Das Ziel der Architektur [MSDS+11] ist das sichere Speichern von Daten in cloudbasierten Speicherstrukturen und der Zugriff darauf durch mobile Endgeräte. Der Nutzer behält durch kryptografische Maßnahmen die Kontrolle über die Daten, da diese auf dem cloudbasierten Speicher nicht im Klartext vorliegen und somit nicht öffentlich einsehbar sind. Der Anbieter des Cloud-Speichers hat zu keiner Zeit Zugriff auf die zu schützenden Daten, weder auf das Schlüsselmaterial, noch auf die Klartextdaten. Dies wird durch eine Verschlüsselung im Endgerät erzielt. Um das Teilen von Daten und somit kollaboratives Arbeiten zu ermöglichen, werden einige Schlüssel auf einem Schlüsselserver abgelegt. Die Übertragung verschlüsselter Schlüssel und Informationen bezüglich des Gruppenmanagements wird in der *Sec*^{2}-Architektur mit dem in der vorliegenden Arbeit vorgestellten erweiterten SAML Framework umgesetzt.

Die Daten bestehen aus Dokumenten, die im XML Format vorliegen. Durch die Verwendung von XML-Encryption wird es ermöglicht, sowohl ganze Dokumente als auch nur Dokumententeile zu verschlüsseln. So können z.B. für eine Suche auf verschlüsselten Daten unverschlüsselte

```
<?xml version="1.0" encoding="utf-8"?>
<samlp:Response xmlns:samlp="...:SAML:2.0:protocol"
 xmlns:saml="...:SAML:2.0:assertion"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 ID="response1" Version="2.0" InResponseTo="query1">
  <saml:Issuer>[ID des KS]</saml:Issuer>
  <samlp:Status>
    <samlp:StatusCode Value="...:SAML:2.0:status:Success" />
  </samlp:Status>
  <ds:Signature>
    <!-- Signatur analog zum Request-->
  </ds:Signature>
  <saml:Assertion ID="assertion1" Version="2.0">
    <saml:Issuer>[ID des KS]</saml:Issuer>
    <saml:Subject>[ID des KR]</saml:Subject>
    <saml:Conditions NotBefore="[Timestamp]" NotOnOrAfter="[Timestamp]" />
    <saml:AttributeStatement>
      <saml2:EncryptedAttribute>
        <!-- Anfang des verschlüsselten Inhalts -->
        <ds:KeyInfo>
          <ds:KeyName>Key-ID, Key-Name, o.ä.</ds:KeyName>
          <xenc:EncryptedKey>
            <xenc:EncryptionMethod Algorithm=".../xmlenc#rsa-oaep-mgf1p" />
            <xenc:CipherData>
              <xenc:CipherValue>
               [Verschlüsselter Schlüssel, der vom KR angefordert wurde]
              </xenc:CipherValue>
            </xenc:CipherData>
          </xenc:EncryptedKey>
        </ds:KeyInfo>
        <!-- Ende des verschlüsselten Inhalts -->
      </saml2:EncryptedAttribute>
    </saml:AttributeStatement>
  </saml:Assertion>
</samlp:Response>
```

Abb. 6: XML Darstellung einer `Response`

Metadaten hinzugefügt werden. Die abzusichernden Dokumententeile werden verschlüsselt in der Cloud gespeichert. Damit entfällt ein sonst notwendiges Vetrauensverhältnis bzgl. der Datenvertraulichkeit zwischen Benutzer und Cloud-Anbieter. So kann z.B. ein Unternehmen, das einen eigenen Schlüsselserver betreibt, die Vorteile der Cloud nutzen und gleichzeitig die Vertraulichkeit seiner Daten sicherstellen.

In *Sec*^{2} wird das zur Ver- und Entschlüsselung notwendige Schlüsselmaterial sowohl auf Client- als auch auf Server-Seite in Hardware-Sicherheits-Modulen abgelegt. Die verschlüsselten Dokumententeile können zwischen mehreren, in Gruppen organisierten, Benutzern geteilt werden. Um diese Anforderung zu erfüllen, werden in der Architektur drei Schlüsselarten verwendet: Der symmetrische Dokumentenschlüssel, der symmetrische Gruppenschlüssel sowie das asymmetrische Benutzerschlüsselpaar, das aus einem privaten und öffentlichen Schlüssel besteht. Der Dokumentenschlüssel verschlüsselt jeweils einen Dokumententeil. Er wird seinerseits mit dem Gruppenschlüssel der Gruppe verschlüsselt, für die der Dokumententeil lesbar sein soll und das resultierende Chiffrat mit dem verschlüsselten Schlüssel im Dokument gespeichert. Die Gruppenschlüssel werden auf einem dedizierten Server abgelegt und auf Anfrage veschlüsselt an den Client gesendet.

Auf der Seite des mobilen Endgerätes wird die Sicherheit der Gruppenschlüssel durch Verwendung einer speziellen SD-Karte (*Secure Flash Card 1.0, Giesecke und Devrient*), die über

kryptografische Funktionen und sicheren Speicher verfügt, gewährleistet. Auf dem Speicher dieser Karte werden die Gruppenschlüssel sowie die privaten Nutzerschlüssel abgelegt. Mit Hilfe der kryptografischen Funktionen können auf der Karte Ver- und Entschlüsselungsoperationen durchgeführt werden. Dies ermöglicht die mit dem öffentlichen Teil des Benutzerschlüsselpaars verschlüsselte Übertragung des Gruppenschlüssels, der somit außerhalb eines sicheren Hardware-Speichers niemals in unverschlüsselter Form vorliegt. Mit dem Gruppenschlüssel können die Ver- und Entschlüsselungen von Dokumentenschlüsseln von der Karte durchgeführt werden. Die Ver- und Entschlüsselung der Teildokumente mit den Dokumentenschlüsseln hingegen erfolgt aus Gründen der Effizienz nicht auf der Karte, sondern auf dem mobilen Endgerät.

4.2 Aufgabe des SAML Frameworks

In der zuvor beschriebenen Architektur wird das SAML-Framework benutzt, um verschlüsselte Schlüssel und Informationen bezüglich des Gruppenmanagements zu übertragen.

Für den Anwendungsfall von SAML im Forschungsszenario sind der auf dem mobilen Endgerät installierte Client sowie der Schlüsselserver relevant. Um Dokumente aus der Cloud ver- und entschlüsseln zu können, stellt der Client Anfragen nach Informationen oder Schlüsseln, die der Server beantworten muss. Die Identifizierbarkeit des Clients muss sichergestellt werden, sodass keine Fälschungen hinsichtlich seiner Identität möglich sind. Des Weiteren muss der Inhalt der Nachricht vor Veränderungen geschützt und die Vertraulichkeit gewährleistet sein.

Für die Sicherheitsmechanismen von SAML ist eine Public-Key Infrastruktur notwendig, da zur Verschlüsselung und Signatur der SAML Nachrichten asymmetrisches Schlüsselmaterial vorhanden sein muss, das entweder beidseitig bekannt sein muss oder dem mit Hilfe einer Zertifikatskette vertraut werden muss. Dies wird im Forschungsprojekt durch die Installation des asymmetrischen Benutzerschlüsselpaars und des öffentlichen Teils des Serverschlüssels auf der Sicherheitskarte im mobilen Endgerät erzielt.

Um ein Teildokument aus der Cloud zu entschlüsseln, müssen zunächst die entsprechenden symmetrischen Dokumentenschlüssel mit dem Gruppenschlüssel entschlüsselt werden. Wenn der Gruppenschlüssel nicht schon im Besitz des Clients ist, muss er ihn vom Schlüsselserver abrufen. Dazu stellt der Client eine SAML Anfrage gemäß dem in Abschnitt 3 vorgestellten Schema, in der die Informationen zum Gruppenidentifikator als Schlüsselreferenz enthalten ist. Der Schlüsselserver beantwortet die Anfrage, falls sie berechtigt ist, mit einer Response, die eine SAML Assertion mit dem Gruppenschlüssel enthält. Somit ist der übertragene Schlüssel durch die Signatur authentifiziert und seine Vertraulichkeit ist durch die Verschlüsselung sichergestellt. Gruppenschlüssel werden ausschließlich auf der Karte entschlüsselt und verlassen diese niemals.

Die administrativen Anfragen an den Schlüsselserver, die per SAML gesichert übertragen werden, umfassen folgende Punkte, die sich in zwei Kategorien unterteilen lassen.

1. Informationsabfragen, die den Zustand des Systems nicht verändern:
    1. Abruf von Informationen über eine Gruppe (Schlüssel, Name etc.)
    2. Abruf aller Mitglieder einer Gruppe
    3. Abruf aller Gruppen, in denen ein Benutzer Mitglied ist
    4. Abruf von Informationen über einen Benutzer (Email etc.)

5. Abrufen aller Benutzer, mit denen ein Benutzer gemeinsam in Gruppen Mitglied ist

2. Änderungsanfragen, die eine bleibende Änderungen des Zustands des Schlüsselserver hervorrufen:
    1. Erstellen einer Gruppe
    2. Ändern der Attribute einer Gruppe (Name, etc.)
    3. Löschen einer Gruppe
    4. Hinzufügen eines Benutzers zu einer Gruppe
    5. Entfernen eines Benutzers aus einer Gruppe
    6. Ändern der Attribute eines Benutzers (Email, etc.)

Beide Anfragetypen werden in je einer AttributeQuery übermittelt. Bei den Änderungsanfragen sind die Argumente als Kind-Elemente eines Attributs enthalten. Der Server antwortet mit einer Assertion, in der die ordnungsgemäße Ausführung der Änderung bestätigt wird.

Die Implementierung der beschriebenen Funktionalitäten wurde in der Programmiersprache *Java* unter Benutzung des *OpenSAML*-Frameworks [Shib], einer Open Source Biblothek, die SAML Nachrichten verarbeitet, durchgeführt. Durch das vorgestellte Konzept, die Erweiterungspunkte von SAML zu nutzen, konnte zusätzlich zum authentifizierten und vertraulichen Transport von Schlüsselmaterial der Entwicklungsaufwand reduziert werden, da auf bestehende Technologien aufgebaut wird, anstatt eine komplette Neuentwicklung vorzunehmen.

5 Schlussfolgerung

Die vorgeschlagene Erweiterung basierend auf dem SAML Framework stellt eine flexible, typenunabhängige und sowohl von Anwendungen als auch Herstellern unabhängige Lösung für einen gesicherten und authentifizierten Schlüsselaustausch dar. Das Gesamtkonzept ist vollständig SAML kompatibel, so dass Änderungen an bestehenden Infrastrukturen nur zur Implementierung der erweiterten Funktionalität notwendig sind.

Die Umsetzbarkeit des Konzepts wurde bereits im Rahmen eines Forschungsprojektes nachgewiesen und zeigt im Beispielszenario die Vorteile der Lösung auf. Entwickler und Anwender können auf eine neue Lösung zum standardisierten Schlüsselaustausch zurückgreifen und beugen somit Problemen durch eine Vielzahl konkurrierender und gegenseitig inkompatibler Systeme vor.

Literatur

[CDFS+07] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer: OpenPGP Message Format. IETF RFC 4880 (2007).

[CSFB+08] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF RFC 5280 (2008).

[ERHR13] D. Eastlake, J. Reagle, F. Hirsch, T. Roessler: XML Encryption Syntax and Processing Version 1.1. In: *W3C Recommendation* (2013).

[ERSH+13] D. Eastlake, J. Reagle, D. Solo, F. Hirsch, M. Nyström, T. Roessler, K. Yiu: XML Signature Syntax and Processing Version 1.1. In: *W3C Recommendation* (2013).

[FGMF+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee: RFC 2616, Hypertext Transfer Protocol – HTTP/1.1. `http://www.rfc.net/rfc2616.html` (1999).

[HaMy05] P. Hallam-Baker, S. Mysore: XML Key Management Specification (XKMS 2.0). In: *W3C Recommendation* (2005).

[HKHS10] Hardjono, Klingenstein, Howlett, Scavo: SAML V2.0 Kerberos Web Browser SSO Profile Version 1.0. Tech. Rep. (2010).

[JaPS13] T. Jager, K. G. Paterson, J. Somorovsky: One Bad Apple: Backwards Compatibility Attacks on State-of-the-Art Cryptography. *In: Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2013).

[JaSo11] T. Jager, J. Somorovsky: How To Break XML Encryption. *In: Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)* (2011).

[JeLS09] M. Jensen, L. Liao, J. Schwenk: The curse of namespaces in the domain of XML signature. *In: Proceedings of the 2009 ACM workshop on Secure web services*, ACM (2009), 29–36.

[MiTe] Microsoft, Technet: Active Directory Architecture. `http://technet.microsoft.com/en-us/library/bb727030.aspx`.

[MSDS+11] C. Meyer, J. Somorovsky, B. Driessen, J. Schwenk, T. Tran, C. Wietfeld: Sec2 – Ein mobiles Nutzer-kontrolliertes Sicherheitskonzept fur Cloud-Storage. *In: Proceedings of the DACH Security 2011* (2011).

[NYHR05] C. Neuman, T. Yu, S. Hartman, K. Raeburn: The Kerberos Network Authentication Service (V5). IETF RFC 4120 (2005), `http://www.ietf.org/rfc/rfc4120.txt`.

[RHPM+08] N. Ragouzis, J. Hughes, R. Philpott, E. Maler, P. Madsen, T. Scavo: Security Assertion Markup Language (SAML) V2.0 Technical Overview. In: *OASIS standard* (2008).

[Shib] Shibboleth: OpenSAML 2.x. `https://shibboleth.net/products/opensaml-java.html`.

[Tane03] A. S. Tanenbaum: Computernetzwerke. Pearson Studium, 4., Überarb. a. Aufl. (2003).

[web] WebCrypto API. `http://www.w3.org/community/webcryptoapi`.