Christian Czarnecki et al. (Hrsg.): Workshops der INFORMATIK 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 85

Umsetzung einer Architektur zur ad-hoc Etablierung vertrauenswürdiger Kommunikation

Matthias Enzmann[1,2], Dominik Spychalski[1,2]

Abstract: Der Einsatz von Kryptografie ist das Mittel der Wahl, um vertrauliche Kommunikation zwischen zwei Parteien abzusichern. In der Praxis gestaltet sich dies jedoch häufig schwierig, da es in gängigen Anwendungen, wie bspw. E-Mail-Kommunikation, notwendig ist, dass beide Parteien bereits im Vorfeld eine kryptografische Ausstattung (Schlüsselpaar, konfigurierte Software, etc.) besitzen müssen, um vertraulich zu kommunizieren. Dies kann für Neu-Nutzer eine frustrierende Erfahrung werden, da sie sich zunächst um ihre eigene kryptografische Ausstattung kümmern müssen, was für Laien bereits schwierig sein kann, und dann feststellen müssen, dass ihnen diese nicht weiterhilft, wenn ihr Kommunikationspartner nicht auch bereits kryptografisch ausgestattet ist. In diesem Beitrag wird eine Architektur vorgeschlagen, welche sich diesem Problem annimmt. Kryptografisch ausgestattete Nutzer sollen in die Lage versetzt werden, mit Nutzern vertraulich zu kommunizieren, die noch keine kryptografische Ausstattung haben. Dabei soll die bei einer Partei evtl. bereits vorhandene kryptografische Ausstattung auch von der Partei ohne bisherige kryptografische Ausstattung genutzt werden können.

Keywords: Kommunikations-Architektur, E-Mail-Verschlüsselung, Identity-Based-Encryption

1 Einführung und Motivation

In einer digitalen Gesellschaft ist der Austausch von Daten und Informationen zwischen Personen oder Diensten essentiell. Der Anwendungskontext reicht hierbei vom privaten bis hin zum geschäftlichen Bereich. Um Informationen zwischen zwei Parteien, einem Sender und einem Empfänger, auszutauschen, existieren die unterschiedlichsten Ansätze, Protokolle und Standards. Trotz der vielen Möglichkeiten zur Kommunikation ist die E-Mail immer noch das mengenmäßig am häufigsten genutzte Kommunikationsmedium. Die Anzahl der gesendeten E-Mails beläuft sich laut einer Schätzung der Firma Statista aus dem Jahre 2017 auf täglich 269 Milliarden E-Mails (einschließlich geschätzter 148 Milliarden Spam- E-Mails) [ST17].

Zur Absicherung E-Mail-basierter Kommunikation haben sich Standards wie zum Beispiel S/MIME oder PGP etabliert, welche jeweils auf Basis von kryptografischen Verfahren arbeiten [SEH00]. Für den Einsatz von kryptografischen Verfahren zum Ver- und Entschlüsseln von Nachrichten ist es notwendig, dass beide Kommunikationspartner

[1] Fraunhofer-Institut für Sichere Informationstechnologie, Rheinstr. 75, 64295 Darmstadt, vorname.nachname@sit.fraunhofer.de
[2] Dieser Beitrag wurde vom Bundesministerium für Bildung und Forschung (BMBF) sowie vom Hessischen Ministerium für Wissenschaft und Kunst (HMWK) im Rahmen von CRISP gefördert.

im Vorfeld bereits kryptografische Schlüssel besitzen bzw. erzeugt haben und ihre jeweilige E-Mail-Client-Anwendung damit konfiguriert haben. Erst dann sind die Voraussetzungen für eine abgesicherte bidirektionale Kommunikation gegeben. Diese Konstellation stellt jedoch viele Anwender sehr oft vor Probleme, wodurch letztendlich auf einer oder beiden Seiten der Kommunikation die kryptografischen Fähigkeiten fehlen und selbst sensible und sicherheitskritische Informationen deshalb auch im Klartext versendet werden. Das Versenden der Daten im Klartext hat zur Folge, dass diese auf dem kompletten Zustellungsweg der E-Mail mindestens von den E-Mail-Anbietern einsehbar sind. Bei zusätzlich fehlender Transportwegsicherung zwischen den Anbietern, zum Beispiel durch SSL/TLS, reicht einem externen Angreifer der Zugriff auf den Kommunikationskanal, um die Nachricht mitzulesen.

Dieser Beitrag stellt eine Kommunikationsarchitektur vor, welche die Herausforderungen und Probleme der Kommunikationspartner löst und das Etablieren einer vertrauenswürdigen Kommunikation ohne besondere Fähigkeiten ermöglicht.

2 Anwendungsszenario

Das dem Beitrag zugrundeliegende Anwendungsszenario sieht eine Asymmetrie der kryptografischen Fähigkeiten zwischen Sender (Alice) und Empfänger (Bob) vor. Das bedeutet, dass nur eine der beiden Parteien kryptografische Fähigkeiten hat, wie den Zugriff auf kryptografische Software und hierfür konfigurierte kryptografische Schlüssel. Dies schließt den Aspekt ein, dass sich die Kommunikationspartner im Vorfeld nicht bereits auf eine gemeinsame kryptografische Technologie geeinigt haben müssen. Im Folgenden werden zwei Szenarien beschrieben, in denen Alice und Bob vertraulich kommunizieren wollen, die sich im Hinblick auf kryptografische Fähigkeiten unterschieden. Szenario 1 stellt das Basisszenario dar und das darauf aufbauende Szenario 2 das eingangs erwähnte Szenario.

Szenario 1: Weder Alice noch Bob sind im Besitz kryptografischer Schlüssel. Alice erzeugt oder beschafft einen öffentlichen Schlüssel für Bob. Alice verschlüsselt damit die für Bob bestimmte Nachricht und übermittelt diese an ihn. Nach Erhalt der Nachricht beschafft sich Bob den korrespondierenden Entschlüsselungsschlüssel und entschlüsselt die Nachricht. Um Alice vertraulich antworten zu können, führt Bob die gleichen Schritte durch wie Alice zuvor.

Szenario 2: Alice ist bereits im Besitz kryptografischer Schlüssel, welche sie für die Kommunikation nutzen möchte. Bob hat noch keinen kryptografischen Schlüssel. Alice führt die gleichen Schritte aus wie in Szenario 1, verschlüsselt jedoch zusätzlich ihren vorhandenen öffentlichen Schlüssel mit der Nachricht für Bob. Bob verfährt wie in Szenario 1 erhält aber zusätzlich Alice‘ existierenden Schlüssel. Hierdurch ist Bob in der Lage, Alice unter Verwendung ihres existierenden Schlüssels vertraulich zu antworten.

3 Anforderungen

Das Ziel ist es eine Architektur umzusetzen, welche einen bereits mit kryptografischen Mitteln ausgestatteten Sender (Alice) befähigt, vertraulich mit einem Empfänger (Bob) ohne derartige Mittel zu kommunizieren (vgl. [ZHG17]). Zu diesem Zweck soll die Gesamtarchitektur folgende Anforderungen erfüllen:

a) ***Software.*** Da der Empfänger nach Voraussetzung keine kryptografische Software hat, muss diese ad-hoc verteilt und genutzt werden können. Eine Installation von Software im Vorfeld der Kommunikation soll somit nicht notwendig sein.

b) ***Schlüsselerzeugung.*** Die Erzeugung der kryptografischen Schlüssel muss auf eine „sparsame" Art und Weise erfolgen, damit für sie keine besonderen kryptografischen Fähigkeiten notwendig sind. Unter der Voraussetzung, dass die in *a)* beschriebenen Bedingungen erfüllt wurden, kann die Schlüsselerzeugung durch diese Software erfolgen.

c) ***Schlüsselverteilung.*** Jede Partei soll existierende Schlüssel des Kommunikationspartners nutzen können. Insbesondere sollen die durch *b)* erzeugten Schlüssel genutzt werden können.

d) ***Schlüsselzugehörigkeit.*** Die Überprüfung der Schlüsselzugehörigkeit zu einer Partei ist für beide Parteien relevant. Jede, an dem Protokoll teilnehmende Partei muss in der Lage sein, die Bindung zwischen Subjekt (der korrespondierenden Identität) und dem Objekt (dem öffentlichen kryptografischen Schlüssel) zu verifizieren.

4 Architektur

In diesem Abschnitt wird eine Lösungsarchitektur skizziert, welche die zuvor beschriebenen Anforderungen erfüllen kann. Den Grundstein dieser Architektur stellt die so genannte identitätsbasierte Verschlüsselung dar, die im Folgenden kurz erläutert wird.

4.1 Identitätsbasierte Verschlüsselung

Die Idee einer identitätsbasierten Verschlüsselung (*Identity Based Encryption*, IBE) wurde bereits 1985 von Shamir [Sh85] formuliert. Es dauerte jedoch mehr als 15 Jahre bis Boneh und Franklin [BF01] das erste praktische IBE-Verfahren vorstellen konnten.

IBE ist ein Vertreter der so genannten Public-Key-Verfahren, welche einen öffentlichen Schlüssel zur Verschlüsselung und einen privaten Schlüssel zur Entschlüsselung nutzen. Der öffentliche ebenso wie der private IBE-Schlüssel wird in einem IBE-Verfahren von einer gegebenen Identität abgeleitet. Eine Identität stellt in diesem Zusammenhang eine beliebige Bitfolge dar, insbesondere natürlich auch Zeichenketten (*Strings*).

Im Unterschied zu herkömmlichen Public-Key-Verfahren, wie bspw. RSA, werden der öffentliche und private Schlüssel nicht notwendigerweise parallel erzeugt, sondern können zeitlich und räumlich voneinander unabhängig generiert werden.[3] Dies ermöglicht es, dass ein Sender, der für einen Empfänger eine Nachricht verschlüsseln will, den hierfür benötigten öffentlichen Schlüssel alleine durch Angabe der Identität des Empfängers eigenständig erzeugen kann. Die Erzeugungsfunktion des öffentlichen Schlüssels gibt dabei keinerlei Informationen über den dazugehörigen privaten IBE-Schlüssel preis und kann von jedem genutzt werden. Der private Schlüssel wird bei einem IBE-Verfahren nicht vom Besitzer der Identität erzeugt, sondern kann ausschließlich von einer vertrauenswürdigen Schlüsselzentrale, dem so genannten *Key Distribution Center* (KDC), erzeugt werden. Das KDC legt hierfür die kryptografischen Parameter fest, die sowohl für die Erzeugung der öffentlichen wie auch der privaten IBE-Schlüssel verwendet werden. Wer einmal die öffentlichen, kryptografischen Parameter kennt, kann daraus öffentliche Schlüssel für beliebige Identitäten unterschiedlicher Empfänger erzeugen. Empfänger von mittels IBE verschlüsselten Nachrichten müssen sich an das KDC wenden, um den privaten IBE-Schlüssel für die Identität zu erhalten, für die die Nachricht verschlüsselt wurde. Das KDC wird jedoch nur dann den privaten Schlüssel zu einer gegebenen Identität herausgeben, wenn der anfragende Nutzer nachweisen kann, dass er die fragliche Identität inne hat. Wie dieser Nachweis im Einzelfall zu führen ist, ist nicht Teil des IBE-Verfahrens und implementierungsabhängig, da die Art und Weise des Nachweises von der jeweiligen Identität abhängt und den damit zusammenhängenden Möglichkeiten zur Überprüfung.

Zusammenfassend kann ein IBE-Verfahren im Wesentlichen durch die folgenden vier abstrakten Funktionen beschrieben werden:

$\text{Pub}_{\text{ID}} \leftarrow$ *Generate* (ID)	Erzeugt aus dem Bitstring ID einen öffentlichen Schlüssel Pub_{ID}
$\text{Priv}_{\text{ID}} \leftarrow$ *Extract* (ID)	Erzeugt aus dem Bitstring ID einen privaten Schlüssel Priv_{ID} (nur KDC)
$C \leftarrow$ *Encrypt* (Pub_{ID}, M)	Verschlüsselt die Nachricht M mittels Pub_{ID}
$M \leftarrow$ *Decrypt* (Priv_{ID}, C)	Entschlüsselt das Kryptogramm C mittels Priv_{ID} (nur Inhaber der Identität ID)

4.2 Umsetzung der Anforderungen

Die in Abschnitt 3 aufgestellten Anforderungen werden in der hier beschriebenen Architektur wie folgt umgesetzt.

[3] Der inhaltliche Zusammenhang zwischen öffentlichem und privatem Schlüssel, wie bei herkömmlichen Verfahren, ist dennoch gegeben.

a) ***Software.*** Die Software wird als portabler Code (PC) in Form von JavaScript-Programmen entworfen. Dies hat den Vorteil, dass der Code „on-the-fly" verteilt und ausgeführt werden kann.
b) ***Schlüsselerzeugung.*** Die Erzeugung der kryptografischen Schlüssel des Nutzers erfolgt durch den in a) erwähnten portablen Code, sodass keine Vorbereitungen seitens des Nutzers notwendig sind.
c) ***Schlüsselverteilung.*** Die Verteilung der öffentlichen, kryptografischen Schlüssel erfolgt implizit durch die Verwendung von IBE, da jeder Sender den Schlüssel des Empfängers dezentral selbst generieren kann.
d) ***Schlüsselzugehörigkeit.*** Die Überprüfung der Schlüsselzugehörigkeit zu einer adressierten Identität wird durch die Verwendung von IBE vereinfacht. Da der kryptografische Schlüssel direkt aus der adressierten Identität abgeleitet wird, ergibt sich die Bindung des kryptografischen Schlüssels an die Identität unmittelbar, sodass keine weitere Prüfung notwendig wird.

4.3 Rollen

Die Architektur zur Umsetzung der Szenarien aus Abschnitt 2 sieht ein Zusammenspiel verschiedener Akteure vor, denen unterschiedliche Rollen zufallen. Die Rollen sollen im Folgenden zunächst kurz beschrieben werden, bevor ihr Zusammenspiel in Abschnitt 5 erläutert wird.

Identity Provider. Ein Identitätsanbieter (*Identity Provider*, IdP) verwaltet digitale Identitäten für seine Nutzer. Nutzer registrieren sich hierfür beim IdP und legen dort eine oder mehrere Identitäten an. Der IdP stellt sicher, dass angelegte Identitäten eindeutig sind, d.h. unterschiedliche Nutzer nicht dieselbe Identität nutzen können. Hauptaufgabe des IdP ist es, die Authentizität einer gegebenen Identität gegenüber Dritten zu bestätigen. Für eine solche Identitäts-Bestätigung wendet sich der Dritte an den IdP, der den Nutzer dann bspw. per Web-Login authentifiziert und bei Erfolg eine Bestätigung herausgibt, dass der Nutzer die in Frage stehende Identität besitzt. Die Bestätigung erhält der Dritte entweder unmittelbar vom IdP oder mittelbar vom Nutzer in Form eines, zumeist digital signierten, Tokens.

Key Distribution Center. Die Schlüsselzentrale (*Key Distribution Center,* KDC) erzeugt die privaten IBE-Entschlüsselungsschlüssel für Identitäten, die ihr durch einen Nutzer übermittelt werden. Voraussetzung hierfür ist, dass der Nutzer nachweisen kann, dass er die übermittelte Identität inne hat. Wird bspw. der Kontoname eines sozialen Netzwerks wie Facebook als Identität genutzt, nimmt das soziale Netzwerk die Rolle eines Identitätsanbieters an. In diesem Fall muss das KDC die IdP-Schnittstelle jenes Anbieters unterstützen, im Beispiel „Facebook Login", und den Nutzer auf dessen Anmeldeseite weiterleiten, um nach erfolgreicher Anmeldung schließlich die Bestätigung vom IdP zu erhalten, dass der Nutzer die behauptete Identität besitzt.

Message Storage Provider. Der Nachrichtenspeicheranbieter (*Message Storage Provider,* MSP) stellt für Nutzer Speicherplatz zur Verfügung, um verschlüsselte IBE-Nachrichten (Kryptogramme) zu speichern (*Upload*) und abzurufen (*Download*), ähnliche wie ein E-Mail-Server. Der MSP verwaltet jedoch keine Konten und kennt weder die Identität des Nachrichtenerstellers noch die eines Abrufers. Nachrichten werden ausschließlich an Hand einer eindeutigen, zufälligen Nachrichtenkennung adressiert, die beim Upload der Nachricht erzeugt und dem Nachrichtenersteller zurückgesendet wird. Umgekehrt muss ein Abrufer einer Nachricht die Nachrichtenkennung beim Download angeben, um die Nachricht zu erhalten.

Controller. Der Controller (Ctrl) stellt dem Nutzer den Programmcode für die notwendigen Funktionen zum Ver- und Entschlüsseln von Nachrichten sowie deren Up- und Download beim MSP zur Verfügung. Der Programmcode verwaltet die privaten IBE-Schlüssel, die dieser für die verwendeten Nutzeridentitäten vom KDC erhalten hat. Der Controller-Code koordiniert deshalb für den Nutzer auch das Zusammenspiel mit allen Parteien, welche die übrigen Rollen einnehmen. Der Controller-Code wird in der Regel eine Software-Komponente sein, welche lokal auf einem Gerät des Nutzers oder in einer von ihm vertrauten (entfernten) Domäne läuft.

Ein Akteur kann in der Praxis mehr als eine der oben beschriebenen Rollen einnehmen. Dies wird jedoch Auswirkungen darauf haben, wie viel Vertrauen ein Nutzer dem jeweiligen Akteur entgegenbringen muss. Da an jede Rolle bestimmte Erwartungen in puncto Sicherheit geknüpft sind, müssen diese beim Zusammenfallen von Rollen in einem Akteur unabhängig und gleichzeitig von diesem erfüllt werden. Würde bspw. ein KDC zusätzlich als IdP agieren, müsste dem KDC zwar zusätzlich vertraut werden, dass er Nutzeridentitäten selbst sorgfältig prüft, erfordert gegenüber dem KDC aber hinsichtlich der Sicherheit kein nennenswertes Zusatzvertrauen, da ihm ansonsten auch schon hinsichtlich der sorgfältigen Prüfung von Identitätsbestätigungen eines separaten IdP vertraut werden muss, damit private IBE-Schlüssel nur an Identitätsinhaber herausgegeben werden. Anders würde es aussehen, wenn KDC und MSP in einem Akteur zusammenfielen. Denn in diesem Fall müsste zusätzlich darauf vertraut werden, dass der Akteur nicht die privaten IBE-Schlüssel seiner KDC-Rolle nutzt, um Kryptogramme zu entschlüsseln, die ihm in seiner MSP-Rolle anvertraut wurden. Es sind demnach beliebige Rollenkombinationen für einen Akteur denkbar, einige jedoch nur unter starken Vertrauensannahmen, die aus Sicherheitssicht nur bedingt sinnvoll erscheinen.

5 Praktische Umsetzung

Die im vorigen Abschnitt beschriebene Architektur wurde prototypisch im Rahmen eines *Proof-of-Concept* umgesetzt, der in diesem Abschnitt vorgestellt werden soll.

Für beide in Abschnitt 2 beschriebenen Szenarien wird vorausgesetzt, dass Sender und Empfänger existierende E-Mail-Konten besitzen und beide einen Web-Browser auf ihren

jeweiligen Geräten installiert haben, der in der Lage ist, JavaScript-Programmcode zu verarbeiten, bspw. Chrome, Edge, Firefox oder Safari. Für den Sender wird für Szenario (1) angenommen, dass dieser zusätzlich einen E-Mail-Client besitzt, wie bspw. Thunderbird oder Outlook, der verschlüsselte E-Mail-Nachrichten im S/MIME-Format ver- und entschlüsseln kann. Weiterhin wird angenommen, dass der Sender ein herkömmliches Schlüsselpaar besitzt, welches sein E-Mail-Client für die Erstellung verschlüsselter E-Mail-Nachrichten sowie zur Entschlüsselung nutzen kann.

Die Schlüsselzentrale (KDC) ist als Web-Server umgesetzt mit einem REST-Interface zur Abwicklung von Identitätsprüfungen und zum Abruf von privaten Schlüsseln nach erfolgter Identitätsprüfung. Der Nachrichtenspeicher (MSP) ist ebenfalls als Web-Server umgesetzt mit einem einfachen REST-Interface zum Upload und Download von verschlüsselten Nachrichten. Als vereinfachter Identitätsanbieter wird ein E-Mail-Anbieter (*Email Service Provider*, ESP) genutzt.

Für die Umsetzung wurde vereinfacht angenommen, dass der Controller (Ctrl) bzw. dessen portabler Programmcode (PC_{Ctrl}) vom KDC bereitgestellt wird. Dies vereinfacht die Konfiguration des Controller-Code, da bspw. die kryptografischen Parameter des KDC unmittelbar in PC_{Ctrl} konfiguriert werden können. Als Identitäten für die Nutzung von IBE werden E-Mail-Adressen verwendet, da Sender die Adresse von E-Mail-Empfängern ohnehin kennen müssen und somit keine weiteren Annahmen hinsichtlich der verwendeten Identitäten notwendig sind.

5.1 Protokollablauf

In Abb. 1 ist die Architektur des Prototyps sowie die Interaktionen der beteiligten Parteien dargestellt. Der prototypische Ablauf für Szenario (1) würde sich dann wie folgt darstellen.

(1) Sender Alice verfasst Nachricht *M* für Empfänger Bob, erzeugt mittels *Generate* („bob@home.de“) einen IBE-Schlüssel Pub_{Bob} und verschlüsselt *M* mittels *Encrypt* (Pub_{Bob}, *M*) zum Kryptogramm *C*. [Nicht dargestellt in Abb. .]

(2) Alice überträgt *C* an den Nachrichtenspeicher MSP, der eine Referenz „https://msp.de/Ref_C“ auf die gespeicherte Nachricht zurückliefert.

(3) Alice sendet eine E-Mail an Bob, welche schematisch die folgende URL enthält: https://ctrl.de/PC_{Ctrl} #https://msp.de/Ref_C #https://kdc.de/extract #bob@home.de
Dabei ist „https://ctrl.de/PC_{Ctrl}“ die Adresse von der der portable Code des Controller abgerufen wird, „https://msp.de/Ref_C“ die Adresse unter der das zuvor gespeicherte Kryptogramm verfügbar ist, „https://kdc.de/extract“ die URL des KDC und schließlich die Identität „bob@home.de“, die zur Verschlüsselung genutzt wurde.

(4) Bob klickt den Link aus Alice' E-Mail an und sein Browser lädt PC_{Ctrl} vom Controller (Ctrl), d.h. bedingt durch die hier getroffene Vereinfachung vom KDC.

Sobald der Code geladen und gestartet ist, wertet dieser den Rest der URL aus und stellt fest, dass Bob noch keinen Schlüssel für die Identität „bob@home.de“ besitzt. Daraufhin speichert er lokal im Browser zunächst die URL „https://msp.de/Ref_C“ des Kryptogramms und kontaktiert das angegebene KDC unter „https://kdc.de/extract“. In der Anfrage an das KDC übermittelt er die Identität für die der private Schlüssel benötigt wird sowie eine Weiterleitungsadresse „https://ctrl.de/ PC_{Ctrl}/“ (auf sich selbst) für den Fall, dass die Identitätsprüfung erfolgreich verläuft und PC_{Ctrl} den Prozess fortsetzen kann.

(5) Das KDC erhält die Identität „bob@home.de“ sowie die Weiterleitungsadresse „https://ctrl.de/ PC_{Ctrl}/“ und sendet eine E-Mail mit einer Bestätigungs-URL „https://kdc.de/*UID*“ an die angegebene E-Mail-Adresse, wobei „*UID*“ eine zufällig generierte Vorgangsnummer darstellt. Die Weiterleitungsadresse speichert das KDC zur weiteren Verwendung.

(6) Bob ruft die Bestätigungs-E-Mail anschließend ab.

(7) Bob klickt die in der E-Mail enthaltene Bestätigungs-URL, wodurch das KDC kontaktiert wird und feststellen kann, dass die angegebene Identität (=E-Mail-Adresse) unter Bobs Kontrolle ist. Auf Grund dieser positiven Prüfung erzeugt das KDC mittels *Extract* („bob@home.de“) den privaten Schlüssel $Priv_{Bob}$ und löst eine Browser-Umleitung an die in Schritt (5) gespeicherte Adresse „https://ctrl.de/ PC_{Ctrl}/ Ref_{Bob}“ aus. Hierdurch wird PC_{Ctrl} erneut geladen und erhält über die URL eine Referenz Ref_{Bob} auf den privaten IBE-Schlüssel $Priv_{Bob}$. PC_{Ctrl} ruft dann selbstständig den Schlüssel ab und legt ihn im lokalen Browser-Speicher ab.

(8) Im letzten Schritt ruft PC_{Ctrl} noch das Kryptogramm vom MSP ab, mittels der in Schritt (4) gespeicherten Nachrichten-Referenz „https://msp.de/Ref_C“, um aus diesem durch Anwendung von *Decrypt* ($Priv_{Bob}$, *C*) die entschlüsselte Nachricht *M* zu erhalten.

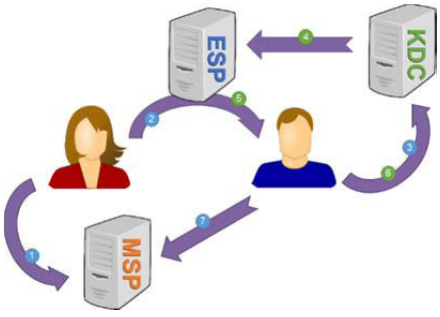

Abb. 1: Grobarchitektur und Abläufe des IBE-Prototypen

Will Bob auf die Nachricht von Alice antworten, müssen die Schritte (1)-(8) mit vertauschten Rollen wiederholt werden. Für den Fall, dass Alice oder jemand anderes

Bob eine weitere mit seiner Identität „bob@home.de“ verschlüsselte Nachricht schickt, stellt PC_{Ctrl} in Schritt (4) fest, dass der private Schlüssel bereits in Schritt (7) eines früheren Programmlaufs abgerufen und gespeichert wurde, sodass er direkt zum finalen Schritt (8) übergehen kann und die Schritte (5)-(7) entfallen.

Der oben geschilderte Ablauf berücksichtigt noch nicht Szenario (2), also den Fall, dass der Sender Alice bereits ein existierendes Schlüsselpaar besitzt, welches der Empfänger Bob für eine Antwort nutzen soll. Hierfür muss Alice in Schritt (1) zusätzlich zur Nachricht *M* noch ihren herkömmlichen öffentlichen Schlüssel PK_{Alice} bzw. dessen zugehöriges Zertifikat Cert(PK_{Alice}) in das Kryptogramm aufnehmen, d.h. *Encrypt* (Pub_{Bob}, *M* || Cert(PK_{Alice})) ausführen, wobei „X || Y“ die Konkatenation von X und Y meint. In Abb. 2 ist in der Oberfläche des Prototypen zu sehen, dass Alice das S/MIME-Zertifikat ihrer „Firmen-Identität“ „/C=DE /O=ACME /OU=People /CN=Alice“ an die für Bob adressierte Nachricht anhängt. Der übrige Ablauf ist identisch, lediglich im letzten Schritt muss PC_{Ctrl} das Zertifikat Cert(PK_{Alice}) noch im Browser-Speicher ablegen, sodass Bob es in seiner Antwort nutzen kann. Antwortet Bob auf Alice’ E-Mail, würde in der gezeigten Oberfläche, nach Eingabe von Alice E-Mail-Adresse „alice@acme.com“ im „To“-Feld, das gespeicherte Zertifikat auftauchen (genauso wie in der Abbildung unter dem „From“-Feld) und der Haken zur Nutzung des Zertifikats automatisch gesetzt. In diesem Fall wird von PC_{Ctrl} kein IBE-Kryptogramm erzeugt, sondern ein CMS-Objekt (Cryptografic Message Syntax), welches sozusagen das Pendant des Kryptogramms im S/MIME-Standard ist. Das CMS-Objekt wird an den Controller geschickt (hier also das KDC), der die S/MIME-Nachricht komplettiert und schließlich als E-Mail an Alice’ Adresse „alice@acme.com“ verschickt. Wenn Alice die E-Mail erhält, sieht diese für sie wie jede andere mit ihrem herkömmlichen Schlüssel PK_{Alice} verschlüsselte E-Mail aus.

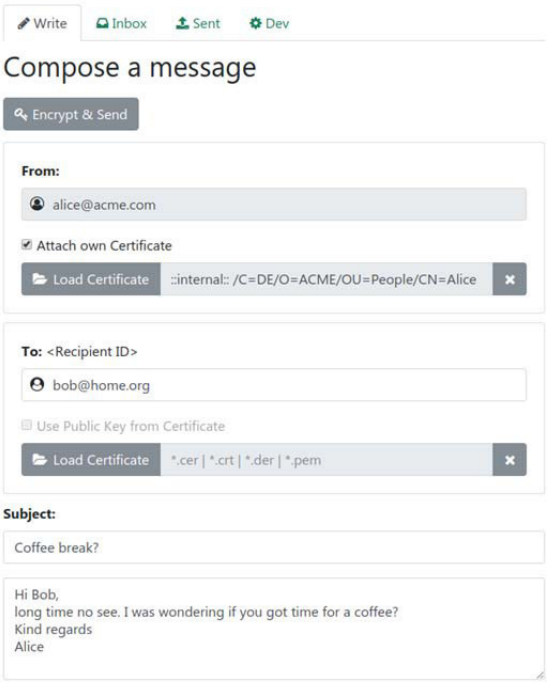

Abb. 2: Oberfläche des Prototypen zum Nachrichtenerstellen

5.2 Sicherheitsaspekte

An dieser Stelle sollen noch einige Aspekte der Sicherheit der oben vorgestellten Anwendung diskutiert werden.

Der Einsatz von identitätsbasierter Verschlüsselung erfordert stets ein besonderes Vertrauen in die Integrität und Sicherheit des KDC, da das KDC jederzeit in der Lage ist, private Nachschlüssel zu erzeugen. Ein korrumpiertes KDC könnte sich somit mit einem MSP verschwören und die vom MSP gespeicherten Kryptogramme von Nutzern entschlüsseln. Dies ist ohne die Annahme, das KDC als vertrauenswürdigen Dritten zu betrachten, kaum zu verhindern. Auf der anderen Seite sind vertrauenswürdige Dritte in Sicherheitsinfrastukturen kein neues Konzept und kommen bspw. in Zertifikatsinfrastrukturen (*Public Key Infrastructure*, PKI) in Form der Zertifizierunsinstanzen (*Certification Authority*, CA) oder bei „*Managed Security*"-Anwendungen häufig vor. Der Aspekt, dass das KDC einen privaten Identitätsschlüssel immer wieder erzeugen kann, hat aber zugleich den Vorteil, dass Nutzer bei Verlust ihres privaten IBE-Schlüssels sich diesen vom KDC stets neu besorgen können.

Die Nutzung von portablem Code in der Art und Weise wie er hier eingesetzt wurde birgt die Gefahr, dass der Code von einem Abruf zum nächsten bösartig verändert wird und die ihm anvertrauten Information nicht mehr in sicherer Art und Weise verarbeitet. Aktuelle Browser können jedoch digitale Fingerabdrücke (Hash-Werte) von durch Web-Seiten eingebundenen Programmcode prüfen und feststellen, ob der von der Web-Seite erwartete Hash-Wert – und somit der erwartete, gegebenenfalls geprüfte Programmcode – mit dem vom Browser berechneten Hash-Wert des geladenen Programmcodes übereinstimmt, sodass der Programmcode nicht ohne Weiteres unbemerkt verändert werden kann. Dies verschiebt das Vertrauen jedoch nur in die Web-Seite, welche den zu prüfenden Hash-Wert für den Programmcode festlegt. Auch hier gilt, dass für den Web-Seiten-Betreiber im Prinzip nichts anderes gilt, als für Hersteller lokal, installierter Software, die sich regelmäßig selbst aktualisiert und von der der Nutzer ebenfalls kaum kontrollieren kann, ob durch Updates ggf. Schadfunktionen in die Software nachgeladen werden.

Die im obigen Ablauf dargestellte Identitätsbestätigung ist sicherlich verbreitet, wenn es bspw. um die Registrierung bei Internet-Diensten geht, bietet aber, was die Sicherheit in den gegebenen Szenarien betrifft, nur einen mäßigen Schutz, da die Bestätigungs-E-Mails des KDC notwendigerweise im Klartext verschickt werden und somit ggf. auf dem Transportweg einsehbar sind. Wird ein „echter“ IdP in das Szenario eingebunden, sollte es beim Identitätsnachweis jedoch keine derartige Lücke geben, da davon ausgegangen werden kann, dass ein „geschäftsmäßiger“ IdP ein sicheres Authentifizierungsprotokoll wie bspw. SAML oder OAuth2 einsetzt.

6 Verwandte Arbeiten

Google bietet für Gmail eine Funktion namens *hosted S/MIME* an, bei welcher die Ver- und Entschlüsselung auf den Mail-Servern geschieht, wodurch dieser die Klartexte mitlesen kann [GO17]. Die Erzeugung der kryptografischen Schlüssel und deren Upload liegt im Verantwortungsbereich des Nutzers. Die United Internet AG bietet für ihren E-Mail-Dienst gmx.de nach der initialen Installation eines externen Browser-Plugin PGP als E-Mail-Verschlüsselung an [GM17]. Auch bei der Erzeugung der kryptografischen Schlüssel hilft die Erweiterung, welche im lokalen Browserkontext des Nutzers abgelegt werden. Beide Ansätze nutzen asymmetrische Kryptografie und setzten voraus, dass sie von beiden Kommunikationspartnern eingesetzt werden und die Schlüssel bereits erzeugt sind. Die Firma MicroFocus bietet eine Integration von *IBE* für ihr Produkt *Voltage Secure Mail* an [MI17], welche unter anderem ein KDC, Erweiterungen für E-Mail-Anwendungen und ein SDK beinhaltet. Der Ansatz setzt das manuelle Installieren der Erweiterung zum Kontakt mit dem KDC und entschlüsseln der Nachrichten voraus.

7 Zusammenfassung und Ausblick

In diesem Beitrag wurde eine Kommunikationsarchitektur vorgestellt, welche es zwei Kommunikationspartnern, die bislang keine kryptografische Ausstattung hatten, erlaubt, vertraulich miteinander zu kommunizieren. Die Architektur unterstützt dabei auch existierende kryptografische Ausstattungen, sodass Nutzer die bereits im Besitz herkömmlicher, kryptografischer Schlüssel sind, anderen Nutzern diese Schlüssel auf einfache Art und Weise zur Verfügung stellen können, sodass Schlüsselinhaber verschlüsselte Nachrichten in gewohnter Art und Weise auch von Nicht-Schlüsselinhabern erhalten können. Die Umsetzung der Architektur nutzt bislang E-Mail-Adressen als Identitäten für das IBE-Verfahren. In zukünftigen Arbeiten sollen Kandidaten für andere Formen von nutzerspezifischen Identitäten identifiziert werden, welche zum einen Sendern „leicht“ zugänglich sind und zum anderen von Empfängern „sicher“ nachgewiesen werden können.

Literatur

[BF01] Boneh, D.; Franklin, M. K.: Identity-Based Encryption from the Weil Pairing. In Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology (CRYPTO ’01), S. 213–229, 2001

[GM17] GMX, Setting up Encrypted Communication, https://support.gmx.com/security/pgp/set-up-encrypted-communication.html, Stand: 30.03.2018.

[GO17] Google, Use hosted S/MIME to keep Gmail messages more secure, https://support.google.com/mail/answer/7023606, Stand: 30.03.2018.

[MI17] MicroFocus, Voltage Secure Mail, https://software.microfocus.com/en-us/product/secure-email-encryption/overview, Stand: 30.03.2018

[Sh85] Shamir, A.: Identity-based Cryptosystems and Signature Schemes. In Proceedings of CRYPTO ’84 on Advances in Cryptology, S. 47–53, 1985

[SEH00] Spychalski, D.; Eckstein, L.; Herfer, M.; Trick, D.; Rubinstein, T.: Die Volksverschlüsselung: Förderung vertrauenswürdiger Ende-zu-Ende-Verschlüsselung durch benutzerfreundliches Schlüssel- und Zertifikatsmanagements. In LNI INFORMATIK 2017, S. 773-785.

[ST17] Statista, Media usage in an internet minute as of July 2017, https://statista.com/statistics/195140/new-user-generated-content-uploaded-by-users-per-minute/, Stand: 01.12.2017.

[ZHG17] Zimmermann, V.; Henhapl, B.; Gerber, N.; Enzmann, M.: Promoting Secure Email Communication and Authentication. In Mensch und Computer 2017 – Tagungsband. Gesellschaft für Informatik. / Workshopband. Gesellschaft für Informatik / Usability Professionals.