Bundesamt
für Sicherheit in der
Informationstechnik

Nutzung von OpenPGP in Webanwendungen

mit Hilfe von Erweiterungen für die Webbrowser Mozilla Firefox und Google Chrome

Autoren

Oskar Hahn (Rechtsanwalt)
Anwälte am Oberen Tor
Obere Straße 30
78050 Villingen-Schwenningen
http://anwaelte-am-oberen-tor.de

Thomas Oberndörfer
Mailvelope GmbH
Schröderstr. 30/1
69120 Heidelberg
https://www.mailvelope.com

Unter Mitwirkung von:

Bernhard Reiter
Emanuel Schütze
Intevation GmbH
Neuer Graben 17
49074 Osnabrück
https://intevation.de

Werner Koch
g10 code GmbH
Hüttenstr. 61
40699 Erkrath
https://g10code.com

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
Tel.: +49 22899 9582-0
E-Mail: bsi@bsi.bund.de
Internet: https://www.bsi.bund.de

Änderungshistorie

Version	Datum	Name	Beschreibung
1.0	11.5.2016	siehe Autoren	Initiale Version für die Veröffentlichung

Inhaltsverzeichnis

1 Einleitung

1.1 Ziel der Studie

In der vorliegenden Studie soll untersucht werden, inwieweit eine Ende-zu-Ende-Verschlüsselung in einer Webanwendung im Webbrowser mittels OpenPGP zum aktuellen Zeitpunkt möglich ist und welche Schritte erforderlich sind, um diese zu verbessern.

Hierfür werden die Anforderungen dargestellt, welche eine OpenPGP-Implementierung für Webseiten erfüllen muss (siehe Kapitel 2). Ein Schwerpunkt liegt auf der Anforderung, dass auch der Betreiber der Webseite nicht in der Lage sein darf, Kenntnis über die Ende-zu-Ende-verschlüsselten Inhalte zu erlangen.

Im Folgenden wird untersucht, inwieweit eine Ende-zu-Ende-Verschlüsselung durch eine Browser-Erweiterung zum aktuellen Zeitpunkt bereits realisiert ist und welche weiteren Entwicklungsschritte erforderlich sind. Hierbei werden zunächst die Schnittstellen von Chrome und Firefox untersucht (Kapitel 3), die vorhandenen OpenPGP-Implementierungen vorgestellt (Kapitel 4) sowie die darauf aufbauenden Browser-Erweiterungen beschrieben (Kapitel 5) und bewertet (Kapitel 6).

1.2 Ausgangssituation

Ende-zu-Ende-Kryptografie per E-Mail ist die verbreitetste Verwendung für das OpenPGP-Protokoll.

Um E-Mails schreiben zu können, werden von den gängigen E-Mail-Providern zwei Wege angeboten. Zum Einen ist es möglich, sich auf seinem Computer einen E-Mail-Client zu installieren, zum Anderen kann ein vom Provider angebotener Webmailer verwendet werden. Während viele installierte E-Mail-Clients auf Desktop-Systemen an die Freie Verschlüsselungssoftware GnuPG angebunden werden können, wird eine Ende-zu-Ende-Verschlüsselung in Webmailern meist nicht angeboten.

E-Mail ist eine gutes Beispiel für eine Ende-zu-Ende-Kryptografie-Webanwendung, weil es die verschiedenen, prinzipiell möglichen Elemente von Krypto-Anwendungen zeigt, für welche eine Nutzung von OpenPGP in Frage kommt.

OpenPGP wird weiterhin verwendet für die Absicherung von

- Installationspaketen auf GNU/Linux-Systemen – uninteressant für die Web-Nutzung.
- Dateien – zukünftig für Web-Ablage-Anwendungen denkbar; in den Eigenschaften vergleichbar mit der Handhabung von Anhängen bei E-Mails.
- Echzeit-Nachrichten – kommt gelegentlich bei XMPP („Jabber“) vor und ist damit auch für Web-XMPP-Klienten denkbar; an Spezifikationen gibt es das veraltete XEP-0027 und neue XEP-Entwürfe. Der Anwendungsfall ist vergleichbar mit den Anforderungen an E-Mail-Nachrichten.

1.3 Ergebnisse

Die Studie hat anhand der Anwendung E-Mail gezeigt, dass eine Ende-zu-Ende-Verschlüsselung im Browser möglich ist, und welche Kooperation von den Anbietern von Web-Diensten nötig ist, um bestimmte Funktionen komfortabel zu implementieren.

Damit vollwertige E-Mails verschlüsselt und entschlüsselt werden können, ist es beispielsweise erforderlich, dass die Browser-Erweiterung mit dem jeweiligen Webmailer über eine API kommuniziert. Von den untersuchten Browser-Erweiterungen stellt bisher lediglich Mailvelope eine solche API bereit. Diese wird bereits durch verschiedene Provider (wie beispielsweise WEB.DE, GMX oder Posteo) genutzt. Weiterhin ist Mailvelope eine der wenigen Erweiterungen, welche wirksame Maßnahmen unternimmt, um den unverschlüsselten Klartext vor dem Zugriff des jeweiligen Mailproviders zu schützen.

Die aktuelle Version von Mailvelope ist nicht in der Lage, GnuPG als Krypto-Backend zu verwenden. Die in GnuPG vorhandenen Schlüssel können nicht automatisch verwendet werden. Weiterhin funktioniert die Key Discovery und Key Authentication nicht optimal.

2 Anforderungen an eine sichere OpenPGP-Implementierung in Webanwendungen

Die Studie beschränkt sich auf den Anwendungsfall, dass ein vertrauenswürdiger Rechner verwendet wird. Nicht untersucht wird der sogenannte Kiosk-Modus, bei welchem ein fremder Rechner (z.B. in einem Internetcafé) genutzt wird.

Es werden drei typische Szenarien betrachtet:

1. Ein Anwender hat an dem Computer an seinem Arbeitsplatz keine Administrationsrechte und kann daher keinen lokalen E-Mail-Client installieren.
2. Ein Anwender hat sich an die Webanwendung seines E-Mail-Providers gewöhnt und möchte deshalb keinen lokalen E-Mail-Client installieren.
3. Ein Anwender hat nicht die erforderlichen technischen Kompetenzen, um sich einen lokalen E-Mail-Client zu installieren und einzurichten.

Bei der Ende-zu-Ende-Verschlüsselung kommt es darauf an, dass die Verschlüsselungssoftware unter der Kontrolle des Anwenders ist und damit lokal vom Anwender installiert wurde. Dies bedeutet, dass die Verschlüsselungssoftware nicht durch die Webanwendung ausgeliefert werden darf. Im Unterschied zu einer lokal installierten Software wird eine Webanwendung bei jedem Besuch der Webseite neu geladen. Der Betreiber der Webseite kann diese daher jederzeit und vom Nutzer unbemerkt verändern. Im Gegensatz zu der Untersuchung einer klassischen Desktop-Software muss bei der Untersuchung einer Webanwendung somit stets zwischen dem Teil der Software unterschieden werden, welcher vom Dienstanbieter geladen wird und dem Teil, der lokal installiert wurde.

Hierbei ist es unerheblich, ob die Software eine Browser-Erweiterung ist oder die Funktionen direkt im Browser integriert wurden. Im zweiten Fall ist die Entwicklung jedoch vom jeweiligen Browser-Hersteller abhängig. Aus diesem Grund erscheint eine Browser-Erweiterung sinnvoller, welche unabhängig vom Browser entwickelt werden kann. Im Folgenden liegt der Schwerpunkt somit auf der Untersuchung von Browser-Erweiterungen. Die im Folgenden aufgeführten Anforderungen können jedoch auch auf eine Integration in den Browser übertragen werden.

In den nachfolgenden Abschnitten werden die Anforderungen (A) für eine sichere OpenPGP-Implementierung beschrieben.

2.1 Absicherung des Klartextes gegenüber der Webanwendung (A1)

Eine Browser-Erweiterung muss sicherstellen, dass die Webanwendung nicht in der Lage ist, an den Klartext der zu verschlüsselnden Daten zu gelangen. Dies ist nur dann gegeben, wenn alle kritischen Schritte durch die Erweiterung und nicht durch die Webanwendung vorgenommen werden. Dies beinhaltet in erster Linie die kryptografischen Funktionen. Weiterhin jedoch auch die Eingabe des zu verschlüsselnden Textes sowie die Schlüsselverwaltung.

Der JavaScript-Code einer Webanwendung ist in der Lage, den DOM der eigenen Webanwendung auszulesen und zu manipulieren. Alle Daten, die direkt in die Webanwendung eingegeben werden, können somit von der Webanwendung abgefangen und an den Server gesendet werden. Aus diesem Grund ist es erforderlich, dass Daten nie im Klartext, sondern nur in verschlüsselter Form, in den DOM eingegeben werden.

Ein anschauliches Beispiel für eine Webanwendung, welche jede Eingabe sofort an den Server überträgt, ist Gmail. Wenn bei Gmail eine neue Nachricht verfasst wird, wird der eingegebene Text regelmäßig an den Server gesendet, um die Nachricht als Entwurf zu speichern. Würde der Nutzer erst den Klartext in das Textfeld eingeben und diesen erst in einem zweiten Schritt verschlüsseln, so wäre der Klartext bereits an die Server von Gmail gesendet worden.

Hieraus folgt, dass die Erweiterung einen eigenen Editor anbieten muss, welcher in einem separaten Fenster, separaten Tab oder innerhalb eines iFrames geladen wird. Erst nach der Verschlüsselung darf der verschlüsselte Text an die Webanwendung übergeben werden.

Das gleiche gilt für Anhänge. Auch diese dürfen nicht direkt in das Dateiupload-Feld der Webseite übergeben werden, sondern müssen vorher durch die Browser-Erweiterung verschlüsselt werden.

Bei der asymmetrischen Verschlüsselung ist es erforderlich, die Zertifikate der Empfänger auszuwählen, bevor die Daten verschlüsselt werden können. Dies geschieht bei lokalen E-Mail-Clients üblicherweise anhand der E-Mail-Adressen der Empfänger. Bei Webmailern besteht die Schwierigkeit darin, dass das Textfeld, in welchem diese E-Mail-Adressen eingetragen werden, nicht normiert ist. Eine Browser-Erweiterung kann somit nicht wissen, aus welchem Textfeld die Empfangsadressen ausgelesen werden können.

Dieses Problem kann dadurch gelöst werden, dass die Browser-Erweiterung eine API anbietet, welche von einer Webanwendungen verwendet werden kann. Hierdurch kann die Webanwendung der Browser-Erweiterung mitteilen, an welche E-Mail-Adressen eine E-Mail versandt werden soll.

Unabhängig davon, wie die jeweilige Erweiterung das Problem löst, muss jedoch sichergestellt sein, dass die Webanwendung dem Nutzer keine Schlüssel unterschieben kann. Dies kann beispielsweise dadurch sichergestellt werden, dass der Anwender vor der Verschlüsselung einen Hinweis erhält, mit welchen öffentlichen Schlüsseln die Daten verschlüsselt werden.

Hieraus folgt, dass die Browser-Erweiterung verschiedene Kontrolldialoge und einen eigenen Editor anbieten muss. Dabei muss der Anwender vor sogenannten Phishing-Angriffen geschützt werden. Bei diesen versucht ein Angreifer eine Webseite so zu gestalten, dass sie identisch zu einer vertrauenswürdigen Webseite aussieht. Gelingt der Angriff bei einem Nutzer, gibt dieser seine schützenswerten Daten auf der gefälschten Webseite ein und übergibt sie somit an den Angreifer.

Diese Anforderung ist schwer umzusetzen, da Webseiten und Browser-Erweiterungen grundsätzlich die selben Web-Technologien zu Verfügung stehen (HTML, CSS, JavaScript) und somit Erweiterungsdialoge von einer Webseite leicht nachgebildet werden können.

Eine mögliche Lösung dieses Problems besteht darin, dass der Hintergrund der Dialoge für jeden Nutzer individuell gestaltet ist. Soweit die individuelle Gestaltung geheim gehalten wird, können die Dialoge und der Editor nicht nachgebildet werden. Ein Phishing-Angriff wird hierdurch erschwert, da er einem aufmerksamen Anwender auffällt.

2.2 Einfacher und sicherer Schlüsselaustausch (A2)

Eine grundsätzliche Problematik bei der asymmetrischen Verschlüsselung ist der Schlüsselaustausch. So kann nur dann eine Nachricht an eine andere Person verschlüsselt versandt werden, wenn der öffentliche Schlüssel der anderen Person bekannt ist.

Herkömmliche Systeme versuchen dieses Problem zu lösen, indem jeder Nutzer seinen öffentlichen OpenPGP-Schlüssel auf einen Keyserver lädt. Andere Nutzer können auf dem Keyserver nach einem Namen oder einer E-Mail-Adresse suchen und so den passenden Schlüssel finden.

Eine weitere Möglichkeit besteht darin, den eigenen öffentlichen Schlüssel bei dem eigenen E-Mail-Provider zu hinterlegen, zum Beispiel im DNS. Dieses Verfahren hat sich aktuell noch nicht durchgesetzt, bietet jedoch im Gegensatz zu Keyservern den Vorteil, dass zu jeder E-Mail-Adresse nur ein öffentlicher Schlüssel hinterlegt werden kann. Während bei Keyservern die Schwierigkeit besteht, den richtigen Schlüssel unter mehreren auszuwählen, werden bei der Hinterlegung von Schlüsseln beim E-Mail-Provider veraltete Schlüssel durch den jeweils Aktuellen ersetzt.

Die Alternative hierzu ist, dass der eigene Schlüssel im Anhang einer E-Mail an die Gegenseite versandt wird und diese den Schlüssel aus dem Anhang in den eigenen Schlüsselbund importiert. Der Vorteil an dieser Variante ist, dass sie auch von technisch weniger versierten Personen nachvollzogen werden kann und keine

weiteren Dienste (wie einen Schlüsselserver oder den E-Mail-Provider) erfordert. Der Nachteil besteht jedoch darin, dass Nachrichten nicht sofort verschlüsselt werden können, sondern zunächst die Schlüssel ausgetauscht werden müssen. Wie im nächsten Abschnitt für andere Schlüsselaustauschmethoden angeführt, kann auch bei angehängtem Schlüssel keine Schlüsselauthentifizierung durchgeführt werden.

Die Problematik des Auffindens des richtigen Schlüssels wird als „Key Discovery“ bezeichnet. Sie ist von der Problematik der „Key Authentication“ zu trennen, bei der es um die Sicherstellung geht, dass der gefundene Schlüssel tatsächlich vom Empfänger erstellt wurde. Dies kann weder durch die Hinterlegung der Schlüssel auf einem Keyserver noch durch eine Hinterlegung beim Provider sichergestellt werden. Beim Keyserver folgt dies daraus, dass Schlüssel von jeder Person hochgeladen werden können. Person A kann einen öffentlichen Schlüssel für Person B veröffentlichen. Bei der Hinterlegung der Schlüssel beim Provider besteht diese Problematik ebenfalls, da dem E-Mail-Provider vertraut werden muss. Wie eingangs dargelegt, kommt es bei der Ende-zu-Ende-Verschlüsselung jedoch darauf an, dass auch dem Provider nicht vertraut werden muss.

Der aktuell noch gängige Lösungsweg für das Problem der Key Authentication ist das sogenannte Web of Trust. Hierbei werden Schlüssel unter Anwesenden überprüft und sodann signiert. Hat man einen Schlüssel von einer noch unbekannten Person erhalten, kann die Authentifikation anhand der bekannten Schlüssel und deren Signaturen vorgenommen werden.

Das Verfahren des Web of Trust ist kompliziert und fehleranfällig. Aus diesem Grund wird in Zukunft mit GnuPG auch das Konzept 'TOFU' unterstützt. TOFU steht für „Trust On First Use“ und bedeutet, dass bei der ersten Nutzung eines Schlüssels der Nutzer die Möglichkeit hat, diesen zu verifizieren. Wird bei einem weiteren Kontakt mit der gleichen Person ein anderer Schlüssel verwendet, wird eine Warnung ausgegeben.

Eine weitere Methode, den Schlüssel der Gegenseite zu authentifizieren, besteht darin, den Fingerabdruck des Schlüssels direkt mit der Gegenseite abzugleichen. Insbesondere bei Privatpersonen, welche die selbe Sprache sprechen, kann beispielsweise die Gegenseite angerufen und der Fingerabdruck telefonisch mitgeteilt werden. Die Software muss hierfür lediglich die Möglichkeit anbieten, den Fingerabdruck des eigenen und des fremden Schlüssels anzuzeigen, sowie am fremden Schlüsseln zu vermerken, dass dieser überprüft wurde.

Auch wenn es wünschenswert wäre, wenn eine Broswer-Erweiterung alle genannten Methoden unterstützt, sind jedenfalls Mindestanforderungen an eine Erweiterung zu stellen. Hiernach muss es möglich sein, den eigenen öffentlichen Schlüssel unkompliziert per E-Mail zu versenden, sowie einen empfangenen Schlüssel zu importieren. In Bezug auf die Authentifikation muss die Erweiterung als Mindestanforderung in der Lage sein, die Fingerabdrücke der verschiedenen Schlüssel anzuzeigen sowie bei fremden Schlüsseln zu vermerken, dass diese überprüft wurden. Bei signierten Nachrichten von anderen Personen muss sodann angezeigt werden, ob der verwendete Schlüssel korrekt ist.

2.3 Einfache Installation (A3)

Aus den eingangs in diesem Kapitel aufgestellten Szenarien ergibt sich die Anforderung, dass die Installation der Verschlüsselungssoftware leicht und ohne Administrationsrechte möglich sein muss.

Je schwieriger die Installation der Erweiterung ist, um so mehr Nutzer werden an dieser scheitern. Aus diesem Grund sollte sich die Erweiterung durch wenige Klicks installieren lassen.

Da sich Erweiterungen für Firefox und Chrome leicht über das Internet mit wenigen Klicks installieren lassen, ist diese Anforderung dann unproblematisch, wenn nur die Erweiterung installiert werden muss. Problematisch ist dies jedoch dann, wenn auf dem PC des Anwenders weitere Software installiert werden muss. Eine solche weitere Software könnte beispielsweise GnuPG sein. Zwar ist die Installation von GnuPG durch das Produkt Gpg4win (der Windows-Portierung von GnuPG) selbst unter Windows relativ einfach, jedoch kann dieser Schritt für technisch weniger versierte Benutzer trotzdem zu schwer sein. Weiterhin setzt die Installation von Gpg4win zum aktuellen Zeitpunkt (Februar 2016) Administrationsrechte voraus. Der zu Beginn des Kapitels genannte Anwendungsfall, dass der Nutzer am Arbeitsplatz keine Administrationsrechte

hat, kann von einer Erweiterung, die zwingend die Installation von GnuPG voraussetzt, derzeit nicht mit Gpg4win gelöst werden.

2.4 Integration mit GnuPG (A4)

Hat ein Anwender bereits die verbreitete Verschlüsselungssoftware GnuPG installiert, besteht die Anforderung, dass sich die Browser-Erweiterung auf GnuPG zurückgreifen kann.

Dies betrifft insbesondere die vorhandene Schlüsselverwaltung von GnuPG. So ist es ein wichtiger Teil der Verwendung von Ende-zu-Ende-Verschlüsselung, dass die öffentlichen Schlüssel der Kommunikationspartner verwaltet werden. Bei Gpg4win wird hierfür beispielsweise der Zertifikatsmanager Kleopatra verwendet.

Bei der Installation einer Browser-Erweiterung sollte diese auf die bestehenden Schlüssel zugreifen können, anstatt die Schlüssel selbst zu verwalten. Diese Integration ist unproblematisch möglich, wenn die Browser-Erweiterung ohnehin auf GnuPG zurückgreift.

Soweit nicht auf GnuPG zurückgegriffen wird, besteht die Anforderung der Erweiterung darin, einen Weg zu finden, wie die vorhandenen Schlüssel komfortabel importiert und bei Veränderungen synchronisiert werden können.

2.5 Universelle Einsetzbarkeit (A5)

Um möglichst allen Internetnutzern Verschlüsselung im Webbrowser anbieten zu können, müssen die gängigen Systeme der Nutzer unterstützt werden.

Die beiden meistgenutzten Browser in Deutschland sind Firefox und Chrome, weshalb eine Erweiterung zumindest diese beiden Browser unterstützen muss.

Weiterhin sollte die Browser-Erweiterung betriebssystemunabhängig sein. Diese Anforderung erscheint daher selbstverständlich, da die Browser Firefox und Chrome auf allen gängigen Desktop-Betriebssystemen verwendet werden können und auch deren Erweiterungen grundsätzlich plattformunabhängig sind. Erweiterungen können jedoch voraussetzen, dass weitere Software auf dem PC des Anwenders installiert ist. In diesem Fall setzt eine universelle Einsetzbarkeit voraus, dass auch diese Software auf den gängigen Desktop-Betriebssystemen (Windows, Mac OS und Linux) lauffähig ist.

Überdies gehört zur universellen Einsetzbarkeit, dass die Erweiterung auf allen Webseiten funktioniert. Soweit beispielsweise E-Mails verschlüsselt werden sollen, ist es nicht ausreichend, dass die Erweiterung nur E-Mails eines bestimmten Providers verschlüsseln kann. Damit sich die Erweiterung an alle richtet, muss die Erweiterung somit in der Lage sein, auf allen Webseiten den einzugebenden Text zu verschlüsseln.

Teil dieser Anforderung ist es auch, dass die Erweiterung mit anderen OpenPGP-Lösungen kompatibel ist. So muss eine Verschlüsselung auch dann möglich sein, wenn der Gesprächspartner Thunderbird mit Enigmail oder Outlook mit GpgOL (aus Gpg4win) verwendet.

2.6 Unterstützung von PGP/MIME (A6)

Zur Verschlüsselung von E-Mails mittels OpenPGP gibt es grundsätzlich zwei Möglichkeiten:

1. Nur der Text einer E-Mail wird verschlüsselt (PGP/INLINE).
2. Der ganze Body einer E-Mail wird verschlüsselt (PGP/MIME).

PGP/INLINE ist ein nicht näher spezifiziertes Verfahren, bei dem der Text einer E-Mail eine mit OpenPGP verschlüsselten Nachricht im ASCII-Format enthält. Der Text der E-Mail wird hierzu verschlüsselt und die binären Daten mit Base64 kodiert, so dass die Daten als ASCII-E-Mail versendet werden können.

Bei PGP/MIME wird tiefer in den Aufbau einer E-Mail eingegriffen. Die Spezifikation findet sich in RFC 3156. Hiernach werden für verschlüsselte und signierte E-Mails spezielle „MIME Content Types“ verwendet, in welche sich die verschlüsselten Daten einer E-Mail befinden. Das die E-Mail im PGP/MIME-Format vorliegt, muss im Header der E-Mail im Feld „Content-Type“ angegeben werden. Der PGP/MIME-Standard sieht auch verschlüsselte Anhänge vor.

2.6.1 PGP/INLINE

Die Vorteile von PGP/INLINE liegen in der Einfachheit. Die verschlüsselte oder signierte Nachricht kann direkt im Text einer E-Mail transportiert werden, ohne dass Besonderheiten einer E-Mail beachtet werden müssen. Aus diesem Grund kann beispielsweise ein Text in einem externen OpenPGP-Client eingegeben und verschlüsselt werden und mittels Copy&Paste in das Textfeld des E-Mail-Providers kopiert werden. Browser-Erweiterungen können somit OpenPGP-Nachrichten versenden, ohne dass eine tiefe Integration mit der Webseite erforderlich ist.

Der Nachteil besteht jedoch darin, dass PGP/INLINE als Standard nicht näher spezifiziert ist und somit spezielle Fälle der Kommunikation via E-Mail nicht abgedeckt werden. Insbesondere gibt es kein einheitliches Verhalten, wie Anhänge einer E-Mail über PGP/INLINE verschlüsselt werden sollen. Weiterhin sieht die Konvention nur reines Textformat und keine HTML-E-Mails vor. Darüber hinaus könnten signierte (unverschlüsselte) E-Mails auf dem Transportweg verändert werden, was zu einer ungültigen Signatur führen würde.

2.6.2 PGP/MIME

PGP/MIME ist in RFC 3156 spezifiziert und wird von den meisten OpenPGP-Clients unterstützt.

Im Gegensatz zu PGP/INLINE unterstützt PGP/MIME Anhänge, wobei auch die Dateinamen der Anhänge verschlüsselt werden. Weiterhin können verschlüsselte E-Mails als HTML kodiert sein.

Der Nachteil besteht darin, dass der Textteil der Nachricht und die Anhänge in der Regel im selben PGP/MIME-Block verschlüsselt werden. Dies hat zur Folge, dass zum Lesen der Nachricht die komplette Nachricht inklusive aller Anhänge heruntergeladen und entschlüsselt werden muss. Dies ist insbesondere in Umgebungen mit begrenzter Download-Bandbreite kritisch.

Weiterhin ist bei PGP/MIME eine tiefere Integration zwischen der Browser-Erweiterung und dem Webmailer erforderlich, da es nicht ausreicht, den verschlüsselten Text zu übergeben, sondern darüber hinaus auch das interne Gerüst der E-Mail entsprechend des Standards aufgebaut werden muss.

2.6.2.1 Schreiben von E-Mails im PGP/MIME-Format

Beim Generieren einer E-Mail im PGP/MIME-Format muss der Header der E-Mail angepasst werden. Üblicherweise ist dies bei einem Webmailer nicht möglich. Die Server-Software von Webmailern erstellen die E-Mail selbst und lassen sich vom Client lediglich den Text übergeben, welcher später in den Body der E-Mail eingesetzt wird.

Eine weitere Möglichkeit besteht darin, dass die Browser-Erweiterung die komplette E-Mail (inklusive Header) selbst erstellt und im Ganzen an den E-Mail-Provider übergibt. E-Mail-Provider bieten in der Regel neben ihrem Webmailer auch einen SMTP-Server zum Senden von E-Mails an.

Diese Möglichkeit hätte jedoch erhebliche Nachteile zur Folge. Die Browser-Erweiterung müsste alle Funktionen eines lokalen E-Mail-Programms anbieten. Insbesondere muss die eigene E-Mail-Adresse als Absender definiert werden und die E-Mail nach dem Versenden in den passenden Gesendet-Ordner kopiert werden. Letzteres ist nur über das IMAP-Protokoll möglich, welches die Browser-Erweiterung somit zusätzlich unterstützen müsste. Die E-Mail müsste bei SMTP über den TCP-Port 25 an den E-Mail-Provider gesendet werden, welcher durch eine Firewall gesperrt sein kann.

Die einzige empfehlenswerte Möglichkeit ist daher, dass die Browser-Erweiterung eine API anbietet, die von den Webmailern genutzt wird. Die Browser-Erweiterung könnte damit die nötigen Informationen vom Webmail-Client erhalten, um eine PGP/MIME-E-Mail über den Webmailer zu versenden. Eine solche API wird bereits von der Browser-Erweiterung Mailvelope angeboten.

PGP/MIME definiert, wie bereits beschrieben, eine äußere MIME-Struktur, die in der Regel vom Server des Mail-Providers erzeugt wird. Der MIME-Knoten, der die verschlüsselten Daten enthält, besteht noch einmal aus einer inneren MIME-Struktur mit Knoten (z. B. für die verschiedenen Kodierungen der Nachricht, HTML oder Text) und zusätzlich einem MIME-Knoten für jeden Anhang.

Die Erweiterung hat daher die Möglichkeit, den PGP/MIME-Knoten zu erzeugen und diesen an den Webmailer zu übergeben. Dieser kann jedoch nicht in das übliche Textfeld eingetragen werden, sondern muss über eine spezielle API übertragen werden.

Der Nachteil an dieser Methode besteht darin, dass die Browser-Erweiterung E-Mails im PGP/MIME-Format nur über Provider verschicken kann, welche die entsprechende API unterstützen.

2.6.2.2 Anzeigen von E-Mails im PGP/MIME-Format

Beim Entschlüsseln einer verschlüsselten Nachricht im PGP/MIME-Format wird nicht der komplette Body einer E-Mail benötigt. Es ist ausreichend, den verschlüsselten Text zwischen

```
-----BEGIN PGP MESSAGE-----
[...]
-----END PGP MESSAGE-----
```

zu verarbeiten. In Abhängigkeit des MIME-Headers „Content-Disposition“ wird der verschlüsselte Text im Webmailer entweder als gewöhnlicher Inhalt einer E-Mail oder als Anhang angezeigt. Im ersteren Fall kann eine Browser-Erweiterung den verschlüsselten Text aus dem DOM des Webmail-Clients auslesen. Soweit der Webmailer den verschlüsselten Text jedoch als Anhang anzeigt, ist eine tiefe Integration von Browser-Erweiterung und Webmail-Client nötig.

Bei einer nur signierten Nachricht ist die Anzeige unproblematisch möglich. Nach RFC 3156 (Abschnitt 5) besteht eine solche E-Mail aus einem separaten Textknoten und einem Knoten für die Signatur. Ein Webmailer zeigt in diesem Fall den Text der E-Mail in der üblichen Weise an und stellt die Signatur als Anhang dar. Problematisch ist jedoch das Verifizieren der Signatur. In diesem Fall reicht es nicht aus, den Text der E-Mail zu überprüfen, da der Header des PGP/MIME-Bodys auch Teil der Signatur ist (siehe RFC 1847, Abschnitt 2.1). Aus diesem Grund kann die Signatur nur bei Webmailern überprüft werden, welche über eine API mit der Browser-Erweiterung kommunizieren und über diese den kompletten MIME-Knoten übertragen.

Bei einer sowohl verschlüsselten als auch signierten Nachricht befindet sich die Signatur innerhalb des verschlüsselten Textes. Aus diesem Grund ergeben sich keine Besonderheiten zu einer nur verschlüsselten E-Mail.

2.6.3 Fazit

Aufgrund der dargestellten Nachteile von PGP/INLINE ist an eine Browser-Erweiterung die Anforderung zu stellen, dass sie PGP/MIME unterstützt. Dies gilt insbesondere für die Anzeige von E-Mails im PGP/MIME-Format, da die Entscheidung, in welchem Format eine E-Mail versandt wird vom Absender getroffen wird.

Da für die Unterstützung von PGP/MIME nicht nur die Erweiterung, sondern auch die Webanwendung entsprechend angepasst werden muss, muss die Erweiterung in der Lage sein, auf anderen Seiten auf PGP/INLINE zurückzufallen.

2.7 Entwicklungsprozess, Entwicklergemeinschaft und Codequalität (A7)

Die Verfügbarkeit der Applikation als Freie oder Open Source Software ist keine feste Anforderung, vereinfacht aber eine Beurteilung erheblich und schafft durch die Möglichkeit unabhängiger Reviews viel Vertrauen in die Qualität der Programmquellen. Ähnliches gilt für einen offenen Entwicklungsprozess. Insbesondere durch einen öffentlichen Bug-Tracker wird sichergestellt, dass Sicherheitsprobleme zeitnah Beachtung finden oder zumindest bekannt sind.

Unabhängig vom Entwicklungsprozess ist es wichtig, dass die Software sich in aktiver Entwicklung befindet und regelmäßige Releases erfolgen, in denen zumindest gefundene Fehler behoben werden.

Um das Vertrauen in die Sicherheit der Software in einer bestimmten Version zu erhöhen, ist es weiterhin förderlich, wenn Security-Audits durch unabhängige Dienstleister durchgeführt werden. Diese können gute Hinweise in Bezug auf die Sicherheit der Software geben. Die dabei gefundenen Sicherheitsprobleme sollten zeitnah behoben und anschließend veröffentlicht werden.

Ein weiterer Weg, die Codequalität und das Vertrauen in korrektes Verhalten der Software zu erhöhen, ist die Nutzung von Unit-Tests. Mindestens für die kryptografischen Algorithmen sollte eine hohe Testabdeckung angestrebt werden.

2.8 Zusammenfassung der Anforderungen

Zusammengefasst bestehen für die Verschlüsselung im Webbrowser folgende Anforderungen:

- **A1:** Die Erweiterung muss eine Umgebung schaffen, in welcher der Nutzer den Klartext seiner Daten eingeben kann, ohne dass die Webanwendung die Möglichkeit hat, den Klartext auszulesen.
- **A2:** Die Erweiterung muss die Probleme der Key Discovery und der Key Authentification lösen.
- **A3:** Die Installation der Erweiterung muss auch für technisch weniger versierte Nutzer möglich sein.
- **A4:** Bei einer bestehenden Installation von GnuPG muss die Erweiterung auf die vorhandenen Schlüssel zugreifen oder komfortabel importieren können.
- **A5:** Die Erweiterung muss die Browser Firefox und Chrome auf Windows, Mac OS und Linux unterstützen und Texte auf allen Webseiten verschlüsseln können.
- **A6:** Die Erweiterung muss PGP/MIME unterstützen.
- **A7:** Der Entwicklungsprozess, die Entwicklergemeinschaft und der Code müssen eine hohe Qualität vorweisen.

3 Analyse der Schnittstellen von Chrome und Firefox

Sowohl Chrome als auch Firefox bieten verschiedene Schnittstellen an, über welche Erweiterungen vom Nutzer aufgerufene Webseiten manipulieren können. Auf diese Weise kann eine Ende-zu-Ende-Verschlüsselung in dem jeweiligen Browser angeboten werden. Neben den hier vorgestellten Methoden stehen den Erweiterungen alle standardisierten JavaScript-APIs zur Verfügung. Insbesondere auch die Web Cryptography API, durch welche eine Erweiterung sichere Zufallszahlen erhalten kann. Bei der Web Cryptography API handelt es sich um eine Empfehlung der W3C, welche von Firefox seit Version 21 und von Chrome seit Version 11 unterstützt wird.

3.1 Chrome

Chrome ist ein Webbrowser aus dem Hause Google. Er wird in Versionen für alle gängigen Betriebssysteme angeboten. Der Großteil des Quellcodes von Chrome wird von Google unter dem Namen Chromium unter einer BSD 3-Klausel Lizenz veröffentlicht.

In Bezug auf die Erweiterungsinfrastruktur sind Chromium und Chrome vergleichbar, weshalb die folgenden Aussagen sowohl für Chrome als auch Chromium zutreffen. Auch ChromeOS, das Betriebssystem von Google, bietet die gleichen Schnittstellen wie der Chrome Browser an. Somit sind Erweiterungen für Chrome auch mit ChromeOS nutzbar.

3.1.1 Chrome Extensions

Die typische Erweiterung für Chrome nennt sich Extension. Eine Chrome Extension wird mit den Web-Technologien HTML, CSS und JavaScript geschrieben. Diese Dateien werden in einem Archiv zusammengefasst und lokal im Browser installiert.

Eine Chrome Extension wird von Chrome in einer Sandbox ausgeführt, welche ihr nur bestimmte Rechte einräumt. Beispielsweise ist es für eine Chrome Extension nicht möglich, ein Pop-Up zu öffnen, ohne das der Nutzer eine Schaltfläche betätigt hat.

Die Sandbox grenzt eine Chrome Extension zu jeder Webseite und zu anderen Extensions ab. Dies führt dazu, dass eine Chrome Extension nur nach Abfrage bestimmter Rechte in den Ablauf einer Webseite oder einer anderen Extension eingreifen kann. Auch ist es nicht möglich, den Speicher (z. B. den local Storage) einer anderen Webseite oder Extension auszulesen. Dieses Konzept wird als Same-Origin-Policy bezeichnet, welches auch dafür verantwortlich ist, dass eine Webseite nicht auf den Speicher einer anderen Webseite zugreifen kann. Im Sinne dieses Konzepts hat jede Erweiterung ihre eigene Herkunft.

Bei der Programmierung einer Chrome Extension kann in einer zentralen Datei definiert werden, dass die Extension in der Lage sein soll, die Sandbox zu verlassen, um beispielsweise auf den jeweils aktiven Tab des Browsers zugreifen zu können. Somit ist Chrome in der Lage, Erweiterungen zu erkennen, welche ein solches Recht benötigen. Der Nutzer wird auf die, von diesen Erweiterungen benötigten Rechte, hingewiesen, wie es beispielsweise auch von Smartphone-Apps bekannt ist. Nur Erweiterungen, bei denen der Nutzer die speziellen Rechte eingeräumt hat, sind in der Lage, auf Webseiten zuzugreifen.

Eine Chrome Extension besteht aus verschiedenen Komponenten, welche unterschiedliche Möglichkeiten bieten. So sind beispielsweise Content Scripts dafür verantwortlich, mit dem Inhalt einer Webseite zu interagieren. Dafür haben diese nur eingeschränkte Zugriffsmöglichkeiten auf die Hauptkomponente der Chrome Extensions, der sogenannten Background Page.

Chrome stellt den Extensions verschiedene APIs zur Verfügung, welche sowohl vordefinierte Funktionen beinhalten als auch Möglichkeiten, mit dem Betriebssystem zu kommunizieren.

Um Daten zu speichern, stehen einer Chrome Extension drei verschiedene Möglichkeiten zur Verfügung:

1. Es besteht die Möglichkeit, eine http-Verbindung zu einem Server aufzubauen und über diese Verbindung die Daten zu übertragen. Die tatsächliche Speicherung der Daten wird in diesem Fall vom Server übernommen. Während JavaScript, das von einer Webanwendung geladen wird, grundsätzlich nur mit der eigenen Domain kommunizieren kann, besteht diese Beschränkung für eine Chrome Extension nicht. Diese kann die Daten an einen beliebigen Server senden. Es ist auch möglich, dass der Server-Dienst auf dem eigenen PC läuft und somit die Daten an eine lokale, von Chrome unabhängige Anwendung, gesendet werden.
2. Weiterhin steht einer Chrome Extension die localStorage API zur Verfügung. Hierbei handelt es sich um eine HTML5-API, welche nicht nur von einer Chrome Extension genutzt werden kann, sondern auch von JavaScript, welches von einer Internetseite geladen wird. Durch den Browser wird dabei nicht ein einheitlicher Speicherplatz zur Verfügung gestellt, sondern ein unabhängiger Bereich für jede Domain. Eine Webanwendung, welche von example1.com geladen wird, kann nicht auf den localStorage zugreifen, welche durch example2.com geschrieben wurde. Die gleiche Regel gilt für eine Chrome Extension, so dass diese ihren eigenen localStorage erhält.
3. Zu guter Letzt bietet Chrome eine Storage API an, welche vergleichbar mit der localStorage API ist, jedoch nur von Chrome Extensions aufgerufen werden kann. Die Unterschiede zwischen der localStorage API und der Storage API sind minimal und werden auf folgender Seite beschrieben.

Jeder der genannten Speichermöglichkeiten ist unverschlüsselt. Hat eine Person Zugriff auf die Festplatte des Nutzers, können die gespeicherten Daten ausgelesen werden. Soweit sensible Daten durch die Erweiterung gespeichert werden sollen, müssen diese zuvor durch die Erweiterung selbständig verschlüsselt werden.

Zu den sensiblen Daten in der Ende-zu-Ende-Verschlüsselung zählen insbesondere die privaten Schlüssel. Diese müssen, anders als der Klartext einer zu verschlüsselnden Nachricht, auf persistente Speicher geschrieben werden. Hierbei stellt sich die Frage, ob der localStorage des Browsers ausreichend ist oder ob der private Schlüssel aus Sicherheitsgründen vom Browser unabhängig auf die Festplatte gespeichert werden sollte.

Aufgrund der hohen Komplexität und notwendigerweise stetigen Entwicklung stellt die Rendering-Engine jedes Browsers eine große Angriffsfläche für Sicherheitslücken dar, mittels derer sich ein Angreifer im schlimmsten Fall alle Rechte des ausführenden Prozesses erschleichen kann. Um diesem Szenario entgegenzuwirken, trennt Chrome die Rendering-Engine jeder einzelnen Website in einen unabhängigen Renderer-Prozess. Dieser läuft als Sandbox und hat damit keinen Zugriff auf umliegende Ressourcen, wie beispielsweise das Dateisystem. Dies gilt jedoch insbesondere nicht für Daten, die im localStorage gespeichert werden, da sich der localStorage innerhalb der Sandbox befindet. Für Daten, die außerhalb der Sandbox liegen (wie geheime Schlüssel, die von GnuPG auf dem lokalen Dateisystem verwaltet werden), bietet dieser Mechanismus aber einen wirksamen zusätzlichen Schutz.

Eine Möglichkeit, dieser Sicherheitsproblematik zu begegnen, besteht darin, den Schlüssel im localStorage mit einem Passwort zu verschlüsseln. Die Sicherheit basiert hierbei auf der Komplexität des Passworts, weshalb Nutzer dazu aufgefordert werden sollten, sichere Passwörter zu verwenden.

Diese Sicherheitsproblematik besteht lediglich für den localStorage und nicht für den Arbeitsspeicher einer Erweiterung. Soweit ein verschlüsselter, privater Schlüssel durch die Erweiterung geladen und entschlüsselt wurde, ist es für einen Angreifer nur dann möglich auf den Schlüssel zuzugreifen, wenn ihm ein Ausbruch aus der Sandbox gelingt.

Da eine Chrome Extension in einer Sandbox läuft, hat sie grundsätzlich keine Möglichkeit, eine Anwendung auf dem lokalen Computer zu starten. Die Ausnahme hierzu bildet Native Messaging. Hierbei kann eine lokal installierte Anwendung definieren, dass sie von Chrome gestartet werden darf. Der Browser und die Anwendung können daraufhin über ein definiertes Protokoll miteinander kommunizieren. Die lokale Anwendung muss jedoch unabhängig von der Chrome Extension installiert werden. Es gibt keine Möglichkeit, die lokale Anwendung aus dem Chrome Web Store zu installieren.

3.1.2 NPAPI

NPAPI steht für Netscape Plug-in API. Es handelt sich um eine Schnittstelle, mit welcher externe Anwendungen in das Verhalten des Webbrowsers eingreifen können. Diese Schnittstelle wird von Chrome seit September 2015 nicht mehr unterstützt und steht somit nicht mehr zur Verfügung.

3.1.3 Native Client

Native Client ist eine Methode, um kompilierte Anwendungen innerhalb von Chrome auszuführen. Hierbei können Programme in C, C++ oder einer anderen Programmiersprache entwickelt werden. Das Programm wird durch einen speziellen Compiler übersetzt, welcher überprüft, dass das Programm keine verbotenen Aktionen ausführt. Innerhalb von Chrome wird das Programm in einer Sandbox gestartet, wodurch die Erweiterung die selben Beschränkungen wie eine Chrome Extension hat.

Durch diese Technologie soll es möglich sein, bestehende Programme bei minimalem Änderungsaufwand als sichere Erweiterung innerhalb des Browsers auszuführen.

Dadurch, dass die Programme innerhalb einer Sandbox laufen, können sie nicht auf das lokale Dateisystem zugreifen. Selbst wenn beispielsweise GnuPG darauf angepasst werden würde, gäbe es keine Möglichkeit, auf vorhandene Schlüssel im Dateisystem zuzugreifen. Weiterhin könnte GnuPG keine Schnittstellen des Betriebssystems verwenden.

Native Client ist somit nicht in der Lage, die Probleme zu lösen, welche bei einer Ende-zu-Ende-Verschlüsselung im Browser auftreten.

Weiterhin wird Native Client nicht von Firefox unterstützt, so dass es keine Browser übergreifende Lösung ist.

Eine Alternative zu Native Client ist das noch in der Entwicklung befindliche Webassembly, welches ähnlich funktioniert, jedoch in allen Browsern (nicht nur in Chrome) laufen soll. Auch hierbei wird der Code innerhalb der Sandbox des Browsers ausgeführt, weshalb die oben genannten Nachteile von Native Client auch auf Webassembly zutreffen.

3.2 Firefox

Der Webbrowser Firefox wird von der Mozilla Foundation entwickelt. Die Codebasis stammt ursprünglich von dem Webbrowser Netscape Navigator. Bei Firefox handelt es sich um eine Freie Software, lizenziert unter der MPL.

Ein Ableger von Firefox ist Iceweasel. Dieser ist aufgrund eines Namensstreits zwischen Mozilla und Debian entstanden, basiert im Wesentlichen jedoch auf Firefox. Die Aussagen dieses Abschnitts treffen somit auch auf Iceweasel zu.

Die Erweiterungsinfrastruktur von Firefox ist historisch gewachsen. Es gibt verschiedene Schnittstellen, die sich von ihrem Funktionsumfang erheblich unterscheiden.

3.2.1 XUL, XPCOM

Aktuelle Erweiterungen für Firefox basieren auf XUL und XPCOM. Hierbei handelt es sich um eine Schnittstelle, über welche eine Erweiterung tief in die Funktionsweise von Firefox eingreifen kann. Firefox bietet verschiedene Möglichkeiten, eine solche Erweiterung zu schreiben. Jedoch hat Mozilla angekündigt, XUL und XPCOM in zukünftigen Versionen von Firefox nicht weiter zu unterstützen. In Zukunft können Erweiterungen somit nicht mehr auf XUL und XPCOM aufsetzen.

Eine neue Erweiterung sollte daher nicht mehr mit XUL und XPCOM entwickelt werden. Eine Ausnahme hiervon gilt für Erweiterungen, welche nur auf Funktionen des Add-ons SDK zugreifen. Obwohl diese aktuell noch auf XUL und XPCOM basieren, hat Mozilla angekündigt, dass diese auch in Zukunft unterstützt werden sollen.

Für die Entwicklung einer auf XUL und XPCOM basierenden Erweiterung bietet Firefox verschiedene Möglichkeiten an:

- Die klassische Variante sind **Overlay Extensions**. Hierbei handelt es sich um XUL-Dateien, welche die Oberfläche von Firefox beschreiben. Eine Erweiterung hat hierbei die Möglichkeit, einzelne Elemente des vollständigen Firefox XUL-Objekts abzuleiten und anzupassen und somit beliebige Funktionalitäten von Firefox zu verändern. Die Funktionalität wird hierbei in JavaScript geschrieben. Da Firefox das XUL-Objekt beim Starten generiert, muss nach einer Installation einer solchen Erweiterung der Browser neu gestartet werden.
- **Restartless Extensions** basieren auf der selben Technologie wie Overlay Extensions, bieten jedoch eine Möglichkeit, dass Firefox nach der Installation der Erweiterung nicht neu gestartet werden muss.
- Das **Add-on SDK** baut auf Restartless Extensions auf, bietet jedoch Funktionen, welche die Entwicklung vereinfachen.

Zwischen diesen Methoden bestehen keine Grenzen. Eine Firefox-Erweiterung kann daher Funktionen von allen Methoden verwenden. Die aktuelle Empfehlung von Mozilla ist, sich auf die Funktionen des Add-ons SDK zu beschränken. In zukünftigen Versionen von Firefox soll nur noch das Add-on SDK unterstützt werden.

Im Gegensatz zu einer Chrome Extension werden Firefox-Erweiterungen, die auf XUL und XPCOM basieren, nicht in einer vergleichbaren Sandbox ausgeführt. Im Ergebnis haben sie die selben Rechte wie eine lokal ausgeführte Anwendung. Sie haben somit nicht nur Zugriff auf den Speicher jeder Webseite, sondern auch von jeder anderen Erweiterung. Ebenfalls kann die Festplatte des Nutzers soweit ausgelesen werden, wie es auch die Software Firefox könnte. Diese Sicherheitsbedenken mögen mit ein Grund gewesen sein, weshalb Mozilla sich entschlossen hat, in Zukunft XUL und XPCOM für Erweiterungen zu deaktivieren. Bis dahin sollten Firefox-Erweiterungen mit der selben Zurückhaltung installiert werden, wie auch bei der Installation einer anderen lokalen Anwendung.

3.2.2 NPAPI

Bei der NPAPI-Schnittstelle von Firefox handelt es sich um die selbe Schnittstelle wie bei Chrome. Doch genau wie bei Chrome wird die Schnittstelle auch von Firefox nicht weiter unterstützt. Aus diesem Grund ist auch unter Firefox NPAPI nicht als Erweiterungsschnittstelle geeignet.

3.2.3 WebExtensions

Mit der Ankündigung, die aktuelle Erweiterungsschnittstelle nicht weiterentwickeln zu wollen, wurde gleichzeitig eine neue Schnittstelle angekündigt. Diese neue Art der Erweiterungen bezeichnet Mozilla als WebExtensions. Diese soll vergleichbar zu den Chrome Extensions sein. Eines der Ziele ist es, dass es einfacher werden soll, Erweiterungen sowohl für Chrome als auch für Firefox zu entwickeln.

Es ist geplant, die Manifest Permissions der Chrome Extensions zu übernehmen, so dass WebExtensions in Zukunft nur dann auf den Speicher von Webseiten zugreifen können, wenn dies durch den Nutzer bei der Installation genehmigt wurde.

WebExtensions können bereits in Nightly, der Entwicklungsversion von Firefox, getestet werden. Mozilla plant, ab Firefox 48 WebExtensions zu nutzen. Diese Version soll am 2. August 2016 veröffentlicht werden.

Native Messaging, die Schnittstelle mit der Chrome Extensions möglicherweise mit GnuPG kommunizieren könnten, soll auch für Firefox umgesetzt werden. Allerdings erst in einer unbestimmten Zukunft.

Zum aktuellen Zeitpunkt kann schwer abgeschätzt werden, wie sich WebExtensions entwickeln werden. Das Ziel, eine browserübergreifende Erweiterungsschnittstelle für Chrome und Firefox zu schaffen, lässt sich mit WebExtensions vermutlich am besten erreichen.

Microsoft hat ebenfalls angekündigt, eine Erweiterungsschnittstelle für den von Browser Edge zu schaffen, die ebenfalls auf Webtechnologien basiert.

3.3 Fazit

Es gibt grundsätzlich zwei Wege, wie eine Browser-Erweiterung entwickelt werden kann, um Ende-zu-Ende-Verschlüsselung im Browser anzubieten.

Die erste Möglichkeit besteht darin, alle erforderlichen Funktionen (inklusive der Schlüsselverwaltung) innerhalb der Erweiterungsinfrastruktur selbst anzubieten. Hierbei liegt es sehr nahe, die Funktionalität in JavaScript zu programmieren. JavaScript ist der gemeinsame Nenner zwischen Chrome und Firefox und kann daher auf diese Weise von beiden Browsern unterstützt werden. In Hinblick darauf, dass Firefox XUL und XPCOM in der Zukunft nicht weiter unterstützen wird, sollte die Ende-zu-Ende-Verschlüsselung so implementiert sein, dass sie nicht auf das lokale Dateisystem zugreifen muss, sondern sich auf den localStorage beschränkt. Auf diese Weise kann die Funktionalität sowohl als Erweiterung für Firefox als auch für Chrome angeboten werden.

Die andere Möglichkeit besteht darin, zusätzlich zu der Browser-Erweiterung eine lokale Anwendung (wie z. B. GnuPG) zu installieren. Die Browser-Erweiterung und die lokale Anwendung müssten bei diesem Ansatz so programmiert sein, dass sie über ein gemeinsames Protokoll kommunizieren. Die Kommunikation könnte beispielsweise über die Native Messaging API geschehen, die bereits von Chrome unterstützt wird und auch in Zukunft durch Firefox unterstützt werden soll. Alternativ könnte die lokale Anwendung einen lokalen TCP-Port öffnen, auf welchen die Browser-Erweiterung zugreifen kann.

Der Vorteil der ersten Möglichkeit besteht darin, dass lediglich die Browser-Erweiterung durch den Anwender installiert werden muss. Dagegen ist es bei der zweiten Möglichkeit erforderlich, dass der Benutzer auch die lokale Anwendung installiert. Dies kann gerade für technisch weniger versierte Benutzer dazu führen, dass auf eine Ende-zu-Ende-Verschlüsselung verzichtet wird. Siehe hierzu auch Abschnitt 2.3.

Die zweite Möglichkeit hat dann einen Vorteil, wenn die lokale Anwendung bereits installiert ist. In diesem Fall kann ein bereits vorhandener OpenPGP-Schlüssel für die Browser-Erweiterung verwendet werden. Ebenfalls können die öffentlichen OpenPGP-Schlüssel in der gewohnten Anwendung (z. B. Kleopatra) verwaltet werden. Siehe hierzu auch Abschnitt 2.4.

Im Umkehrschluss bedeutet dies, dass der Nachteil der ersten Methode darin besteht, dass bei der Verwendung einer von GnuPG unabhängigen Erweiterung der Schlüsselbund von GnuPG nicht verwendet werden kann. Dies gilt insbesondere dann, wenn die Browser-Erweiterung in einer Sandbox läuft und technisch keine Möglichkeiten hat, den Schlüsselbund von GnuPG zu öffnen. In diesem Fall müssen die Schlüssel manuell synchronisiert werden.

Der weitere Vorteil der zweiten Methode besteht darin, dass alle Funktionen von GnuPG genutzt werden könnten, wie beispielsweise die Unterstützung von SmartCards. Bei einer von GnuPG unabhängigen Erweiterung müssten diese Funktionen selbst entwickelt werden, soweit dies innerhalb der Sandbox des Browsers überhaupt möglich ist.

Die Vorteile beider Varianten könnten von einer Browser-Erweiterung dadurch kombiniert werden, dass (wie in der ersten Möglichkeit) alle Funktionen inklusive der Schlüsselverwaltung implementiert werden, jedoch optional mit einer lokalen Anwendung kommuniziert werden kann. Auf diese Weise kann die Erweiterung durch einen technisch weniger versierten Benutzer leicht installiert und benutzt werden. Soweit jedoch eine lokale Anwendung (wie GnuPG) installiert ist, kann auf deren Schlüsselbund und Funktionen zurückgegriffen werden. Auf diese Weise kann sowohl die Anforderung A3 als auch A4 gleichzeitig durch eine Erweiterung erfüllt werden. Erkauft würde dieser Weg mit einem erhöhtem Software-Pflegeaufwand, da nun für viele Funktionen zwei Implementierungen existieren, getestet und abgesichert werden müssten.

4 Beschreibung vorhandener OpenPGP-Implementierungen

Browser-Erweiterungen, welche vom Endanwender installiert werden können, implementieren die kryptografischen Funktionen nicht selbst, sondern bauen auf bestehenden Implementierungen auf. Die Erweiterungen, welche in Kapitel 5 und 6 dieser Studie untersucht werden, basieren auf den OpenPGP-Implementierungen OpenPGP.js, End-To-End und GnuPG.

4.1 OpenPGP.js

OpenPGP.js ist eine Implementation des OpenPGP-Protokolls in JavaScript. OpenPGP.js wird auf GitHub als Freie Software entwickelt. Die Software steht unter der LGPL 3.

OpenPGP.js verfolgt das Ziel, ausschließlich in JavaScript programmiert zu sein. Somit bestehen keine Abhängigkeiten zu anderen Anwendungen, die installiert werden müssten:

„The idea is to implement all the needed OpenPGP functionality in a JavaScript library that can be reused in other projects that provide browser extensions or server applications.“ (Quelle: http://openpgpjs.org/)

Da das Produkt offen entwickelt wird, kann der Aktivitätsstatus des Projekts transparent nachvollzogen werden. Im Jahr 2015 wurden mehrere neue Versionen von OpenPGP.js veröffentlicht. An dem Projekt arbeiten verschiedene Entwickler aktiv mit. Der Code wird durch Unit-Tests automatisch getestet.

Die Software wurde bisher zwei Security Audits durch das deutsche Unternehmen cure53 unterzogen. Die Ergebnisse des ersten Audits wurden von OpenPGP.js veröffentlicht und alle gefundenen schweren und mittelschweren Schwachstellen innerhalb eines Monats behoben.

Da OpenPGP.js alle für die Verschlüsselung erforderlichen Funktionen selbst implementiert, ist eine Installation von GnuPG nicht erforderlich.

Zum aktuellen Zeitpunkt (Februar 2016) wird von OpenPGP.js keine Verschlüsselung mittels elliptischer Kurven unterstützt. Eine Implementierung ist von den Entwicklern jedoch geplant.

4.2 End-To-End

End-To-End wurde im Juni 2014 von Google der Öffentlichkeit vorgestellt. Das ursprüngliche Ziel war es, eine Browser-Erweiterung zu entwickeln, mit der Gmail-Nutzer OpenPGP verwenden können. Mittlerweile liegt der Fokus jedoch mehr auf der Entwicklung einer Ende-zu-Ende-Bibliothek, welche daher auch von anderen Erweiterungen genutzt werden kann. End-To-End steht unter der Apache License 2.

End-To-End ist wie OpenPGP.js komplett in JavaScript implementiert. Hierzu setzt Google nicht auf bereits bestehende JavaScript-Krypto-Routinen auf, sondern hat den kompletten Umfang von OpenPGP neu implementiert. Als Besonderheit nutzt End-To-End die hauseigene Closure Library und den dazugehörigen Closure Compiler, durch dessen Möglichkeit der Typisierung von Variablen und Funktionen Fehler im Quellcode schneller gefunden werden können.

End-To-End unterstützt bereits ECC nach RFC 6637 und nach EdDSA for OpenPGP. Bei der Erstellung von neuen Schlüsseln werden ausschließlich ECC-Schlüssel generiert. Vorhandene RSA-Schlüssel können jedoch verwendet werden.

Der Status der End-To-End-Bibliothek wird von Google als mit sehr ausgereift angegeben. End-To-End ist Teil des Vulnerability Reward Program von Google und wird in Google-Produkten bereits produktiv eingesetzt.

4.3 GnuPG

Bei GnuPG handelt es sich um eine Freie Softwareimplementierung des OpenPGP- und anderen Standards. GnuPG ist eine klassische Desktop-Anwendung, kann auf allen gängigen Betriebssystemen installiert werden und ist lizenziert unter der GPL 3 oder neuer.

Im Gegensatz zu den oben genannten Implementierungen läuft GnuPG nicht innerhalb des Webbrowsers, sondern als eigenständige Anwendung. Aus diesem Grund können Browser-Erweiterungen nicht direkt auf die Funktionen von GnuPG zurückgreifen, sondern müssen über die Erweiterungsschnittstellen des Browsers mit GnuPG kommunizieren (z. B. über Native Messaging). Weiterhin muss GnuPG gesondert installiert werden, wodurch die Installation der Erweiterung erschwert wird.

Der Vorteil besteht jedoch darin, dass GnuPG nicht durch die Sandbox des Browsers beschränkt wird und auf das lokale Dateisystem und Funktionen des Betriebssystem zugreifen kann.

Soweit GnuPG bereits durch den Anwender verwendet wird, besteht der womöglich größte Vorteil von GnuPG gegenüber den anderen OpenPGP-Implementierungen darin, dass die Schlüsselverwaltung von GnuPG verwendet werden kann. Basiert eine Browser-Erweiterung beispielsweise auf OpenPGP.js, so müssen alle Schlüssel aus GnuPG manuell exportiert, in die Erweiterung importiert und bei jeder Veränderung manuell synchronisiert werden.

5 Beschreibung vorhandener Browser-Erweiterungen

Aktuell gibt es verschiedene Browser-Erweiterungen, welche eine Ende-zu-Ende-Verschlüsselung in Firefox und Chrome anbieten.

Für die Untersuchung wurden die Erweiterungen ausgewählt, welche durch die Suche im Chrome Web Store und unter addons.mozilla.org unter den Stichwort PGP gefunden werden konnten und laut ihrer Beschreibung dem Nutzer ermöglichen, Texte im Webbrowser mittels OpenPGP zu verschlüsseln. Weiterhin wird die Erweiterung End-To-End untersucht, welche von Google und Yahoo angeboten wird.

Erweiterung	Entwickler	Basiert auf	Unterstützung für Chrome & Firefox	Lizenz	Letztes Release
Mailvelope	Mailvelope GmbH	OpenPGP.js	ja	AGPL 3	1.3.6 (24.02.2016)
PGP Anywhere	James Cullum	OpenPGP.js	nur Chrome	GPL 2	3.3.0 (24.02.2016)
Mymail-Crypt for Gmail	Sean Colyer	OpenPGP.js	nur Chrome	GPL 2	37 (26.09.2015)
Secure Email	Secure Group	GnuPG	ja	proprietär	5.0.15355.6 (23.12.2015)
Enlocked	Enlocked Inc	unbekannt	nur Chrome	proprietär	3.0.6 (03.09.2014)
WebPG	Kyle L. Huff	GnuPG	ja	GPL 2+	0.9.2.1 (03.01.2013)
End-To-End	Google und Yahoo	End-To-End	nur Chrome	Apache License 2	Development

5.1 Mailvelope

Mailvelope ist ein Browser-Erweiterung, welche sowohl für Chrome als auch für Firefox entwickelt wird. Die Entwicklung startete Anfang 2012 durch Thomas Oberndörfer, der auch Mitautor dieser Studie ist. Mailvelope wird seitdem auf GitHub als Freie Software weiterentwickelt und ist lizenziert unter der AGPL 3.

Im Januar 2015 gründete Thomas Oberndörfer die Mailvelope GmbH zu dem Zweck, Dienstleistungen und Support für Mailvelope anzubieten.

Mailvelope wird von verschiedenen deutschen Unternehmen verwendet. Seit 2014 besteht eine Kooperation mit dem Internetdienstanbieter 1&1 Internet SE. Diese Zusammenarbeit führte dazu, dass die Internetportale GMX und WEB.DE seit August 2015 Mailvelope über eine API unterstützen. Im März 2015 kündigten die Deutsche Telekom und United Internet an, mit Hilfe von Mailvelope eine OpenPGP-Verschlüsselung für De-Mail anbieten zu wollen. Die API von Mailvelope ist offen und kann auch von anderen Anbietern implementiert werden. Die Dokumentation der API findet sich auf: https://mailvelope.github.io/mailvelope/

Weiterhin wird die API von Mailvelope durch den Mailanbieter posteo.de und die frei verfügbaren Webmailer tine20 und Roundcube unterstützt.

Mailvelope setzt auf OpenPGP.js auf. Alle Funktionen zum Erstellen und Verwalten der OpenPGP-Schlüssel werden innerhalb des Browsers ausgeführt. Eine Anbindung an eine lokale Anwendung wie GnuPG ist nicht möglich. Allerdings können private Schlüssel, die mit GnuPG erstellt wurden, in Mailvelope importiert und auf diese Weise genutzt werden.

Der private Schlüssel des Nutzers wird durch ein Passwort verschlüsselt und im localStorage des Browsers gespeichert.

Texte, die mit Mailvelope verschlüsselt werden sollen, werden nicht in die Textfelder der Webseite eingegeben, sondern in einen speziellen Editor von Mailvelope. Auch beim Entschlüsseln einer E-Mail wird der Klartext der E-Mail nicht direkt in den DOM der Webseite eingebaut, sondern in einem iFrame verwendet. Hierdurch ist es der Webanwendung nicht möglich, auf den Klartext der Nachricht zuzugreifen.

Mailvelope bietet einen weiteren Schutzmechanismus: Jede Nutzeraktion (Texteingabe, Klicks) im Editor von Mailvelope erzeugt eine Reaktion in einem Icon in der Add-on Toolbar des Browsers. Es wird für kurze Zeit ein grünes „Ok“ Zeichen eingeblendet und somit dem Nutzer versichert, dass die Eingabe Mailvelope erreicht hat. Falls ein Angreifer die Eingabe dadurch abzufangen versucht, dass das iFrame von Mailvelope durch ein Overlay überlagert wird, hat dieser keine Möglichkeit, die Nutzeraktionen an Mailvelope weiterzuleiten. Somit kann auch ein solcher Angriff dem Nutzer auffallen.

Sowohl der Editor von Mailvelope als auch das im DOM eingebettete iFrame verwenden einen speziellen Hintergrund, welcher vom Nutzer individuell festgelegt wird. Hierdurch ist es einem Angreifer nicht möglich, das Aussehen von Mailvelope nachzubilden, was Phishing-Angriffe erheblich erschwert.

Auch wenn der Name darauf schließen lässt, dass Mailvelope für Webmailer entwickelt wurde, lassen sich mit Mailvelope Texte auf beliebigen Seiten verschlüsseln und anzeigen. Soweit die Webseite jedoch eine spezielle API unterstützt, kann beispielsweise der Empfangsschlüssel durch die Webseite automatisch ausgewählt werden. Weiterhin ermöglicht die API, dass E-Mails auch im PGP/MIME-Format versandt werden können und sich damit auch Anhänge verschlüsseln lassen.

5.2 PGP Anywhere

Bei PGP Anywhere handelt sich um eine Freie Software, lizenziert unter der GPL 2, welche auf GitHub entwickelt wird. PGP Anywhere basiert auf OpenPGP.js.

Zum Zeitpunkt der Untersuchung (Februar 2016) besteht PGP Anywhere aus 11 Commits, welche zum großen Teil am 7. Juli 2015 auf GitHub veröffentlicht wurden. Die Software scheint daher nicht aktiv entwickelt zu werden.

Bei PGP Anywhere handelt es sich um eine Chrome Extension, die daher nicht mit Firefox ausgeführt werden kann. Die Software ist nicht dokumentiert und auch die Oberfläche ist nicht selbsterklärend. OpenPGP-Schlüssel können in PGP-Anywhere erstellt sowie vorhandene Schlüssel importiert werden.

Die Erweiterung bindet sich nicht in Textfelder einer Webseite ein, sondern bietet lediglich ein von der aktuellen Seite unabhängiges Textfeld an. Über dieses Textfeld kann Text verschlüsselt oder entschlüsselt werden. Der verschlüsselte Text muss anschließend manuell mit Copy&Paste in die aufgerufene Webseite übertragen werden.

Der Vorteil an diesem Vorgehen ist, dass hierdurch das JavaScript der aufgerufenen Seite nicht auf den eingegebenen Klartext zugreifen kann. Jedoch führt die fehlende Integration auch zu einem fehlenden Komfort.

5.3 Mymail-Crypt for Gmail

Mymail-Crypt for Gmail ist eine Erweiterung, welche als Freie Software unter der GPL 2 auf GitHub entwickelt wird. Sie basiert auf OpenPGP.js.

Die Erweiterung steht nur für Chrome und nicht für Firefox zur Verfügung. Weiterhin wird ausschließlich die Seite gmail.com unterstützt. Textfelder auf anderen Webseiten können mit der Erweiterung weder verschlüsselt noch entschlüsselt werden. Die Steuerelemente zum Ver- und Entschlüsseln von E-Mails sind direkt in Gmail integriert. So befindet sich beispielsweise das Steuerelement zum Verschlüsseln einer E-Mail direkt unter dem Editor, in welchem die E-Mail geschrieben wird.

Der zu verschlüsselnde Text muss vor dem Verschlüsseln in das von Gmail angebotene Textfeld eingetragen werden, so dass Gmail auf den Klartext zugreifen kann. Dies ist bei Gmail besonders fatal, da der eingetragene Text in kurzen Abständen als Entwurf gespeichert wird. Gmail hat somit zu jeder Zeit Zugriff auf die später zu verschlüsselnden Inhalte.

Beim Testen der Erweiterung im Januar 2016 war es nicht möglich, E-Mails tatsächlich zu verschlüsseln. Ein entsprechender Problembericht ist seit Oktober 2015 auf GitHub gemeldet.

5.4 Secure Email

Secure Email ist ein Teil von dem Produkt Secure Pack, welches von dem Unternehmen Secure Group entwickelt wird. Es handelt sich hierbei um ein proprietäres Softwareprodukt. Die Erweiterung wird für Firefox und Chrome entwickelt.

Neben der Browser-Erweiterung Secure Email gehören zu dem Software Secure Pack verschiedene Anwendungen, um E-Mails auf verschiedenen Smartphone-Betriebssystemen sowie in Outlook zu verschlüsseln. Hierbei liegt der Fokus auf Microsoft Exchange Konten.

Die Browser-Erweiterung Secure Email verwendet als OpenPGP-Implementierung GnuPG. Somit ist es nach Installation der Erweiterung erforderlich, auch GnuPG zu installieren. Die Verwendung von Secure Email ist nur auf Windows möglich. Durch die Integration der Erweiterung zu einer installierten GnuPG-Instanz ist es möglich, alle vorhandenen Schlüssel zu verwenden.

Um Secure Email nutzen zu können, ist ein spezielles Konto erforderlich. Dieses kann nur über die Smartphone-App erstellt werden. Aus diesem Grund ist es für die Verwendung der Browser-Erweiterung erforderlich, dass ebenfalls die Smartphone-App installiert wird. Der erstellte Account muss zunächst von Secure Group freigeschaltet werden, was im Test mehrere Tage dauerte.

Durch die Erweiterung ist es jedoch nur möglich, E-Mails auf einer Outlook Web App zu ver- und entschlüsseln. Auf anderen Seiten lässt sich die Erweiterung nicht verwenden.

5.5 Enlocked

Enlocked ist ein kostenpflichtiger Service, über welchen verschlüsselte E-Mails versandt werden können. Mit einem kostenlosen Account ist der Versand von 10 E-Mails pro Monat möglich. Es handelt sich bei Enlocked somit nicht nur um eine Erweiterung.

Das Grundkonzept von Enlocked besteht darin, dass der zu verschlüsselnde Text einer E-Mail als Anhang versandt wird. Dieser Anhang kann auf eine Homepage von Enlocked geladen werden, wo er entschlüsselt wird. Dieser Schritt findet nach Angaben des Anbieters ausschließlich im lokalen Browser statt, ohne, dass hierfür eine Browser-Erweiterung installiert werden müsste. Wie in Kapitel 2 beschrieben, besteht hierbei die Gefahr, dass der sicherheitsrelevante Quellcode bei jedem Request geändert werden könnte.

Damit E-Mails mittels Enlocked verschlüsselt werden können, muss sowohl der Sender als auch der Empfänger einen Account bei Enlocked haben.

Des Weiteren bietet Enlocked eine Browser-Erweiterung für Chrome an, welche ausschließlich für den E-Mail-Anbieter Gmail funktioniert.

5.6 WebPG

WebPG ist eine Erweiterung für Chrome und Firefox, welche auf GitHub unter der GPL 2 oder neuer entwickelt wird.

WebPG basiert nicht auf einer JavaScript-Implementierung von OpenPGP, sondern verbindet eine lokale GnuPG-Installation mit dem Browser.

Die Erweiterung wird für Firefox und Chrome unabhängig entwickelt, weshalb es sich streng genommen um zwei verschiedene Softwareprodukte handelt.

5.6.1 WebPG-Firefox

Die Erweiterung für Firefox basiert auf NPAPI. Wie in Abschnitt 3.2.2 dargestellt, ist diese Schnittstelle in der aktuellen Version 44 von Firefox zwar noch vorhanden, soll jedoch in zukünftigen Versionen deaktiviert werden. Somit lässt sich die Erweiterung derzeit noch verwenden, ist jedoch auf der aktuellen Codebasis nicht zukunftsfähig.

Die letzte öffentliche Codeänderung an der Erweiterung geschah im Juni 2014. Sie kann daher keine aktive Entwicklung vorweisen.

Die Erweiterung greift nicht auf die Schlüsselverwaltung von GnuPG zurück, sondern versucht diese selbst zu implementieren. Da die Erweiterung jedoch auf GnuPG zurückgreift, können die vorhandenen Schlüssel verwendet werden. Hierbei werden in der Schlüsselverwaltung die in GnuPG gepflegten privaten Schlüssel erfolgreich aufgelistet. Dagegen braucht die Erweiterung bei der Auflistung der öffentlichen Schlüssel mehrere Minuten. In dieser Zeit ist Firefox eingefroren, wodurch diese Funktion nicht wirklich verwendet werden kann.

Über die Erweiterung kann jedes Textfeld im Browser ausgewählt und der eingegebene Text verschlüsselt werden. Hierbei wird kein separater Editor verwendet, so dass das JavaScript der Webanwendung zugriff auf den Klartext hat, bevor dieser verschlüsselt wird.

Verschlüsselter Text soll über einen Kontextmenüeintrag entschlüsselt werden können. Dies hat beim Test der Erweiterung jedoch nicht funktioniert.

5.6.2 WebPG-Chrome

Die aktuelle Version von WebPG-Chrome basiert ebenfalls auf NPAPI. Da Chrome im Gegensatz zu Firefox NPAPI bereits abgeschaltet hat, kann die Erweiterung in Chrome nicht mehr verwendet werden.

Laut einem Issue versuchen die WebPG-Entwickler die Erweiterung auf Native Messaging umzustellen. Hierdurch könnte auf GnuPG zugegriffen werden, ohne NPAPI zu verwenden.

5.6.3 End-To-End

Die bereits im Abschnitt 4.2 vorgestellte OpenPGP-Implementierung End-To-End von Google beinhaltet auch eine Browser-Erweiterung für Chrome. Wie auch die OpenPGP-Implementierung wird die Browser-Erweiterung auf GitHub entwickelt und steht unter der Apache License v2.

Wie bereits dargestellt, liegt der Fokus von Google nicht auf der Browser-Erweiterung, sondern auf der Entwicklung der OpenPGP-Bibliothek. Die Erweiterung befindet sich in einem Alpha-Stadium, im welchem sie nicht produktiv eingesetzt werden sollte. Die Erweiterung kann nicht über den Chrome Web Store installiert werden, sondern muss manuell kompiliert und über eine Entwickleroption in Chrome aktiviert werden.

Ähnlich wie PGP Anywhere integriert sich die Erweiterung zum aktuellen Zeitpunkt nicht in eine aufgerufene Webseite, sondern bietet lediglich ein Icon in der Add-on-Leiste von Chrome an. Dieses Icon öffnet ein Textfeld, über welches Text verschlüsselt und signiert werden kann. Soweit es sich bei dem aktiven Tab um Gmail.com handelt, bietet die Erweiterung die Möglichkeit, den verschlüsselten Text direkt in die Webseite zu kopieren. Auf diese Weise können E-Mails im PGP/INLINE-Format geschrieben werden. PGP/MIME ist nicht möglich.

Neben der von Google angebotenen End-To-End-Erweiterung stellt auch Yahoo eine Browser-Erweiterung für Chrome unter dem selben Namen zur Verfügung. Der Code von Yahoo basiert auf dem Code von Google und wird ebenfalls auf GitHub unter der Apache License veröffentlicht. Im Gegensatz zu Google liegt der Fokus von Yahoo auf der Erweiterung. Diese soll es ermöglichen, sich in den Webmailer von Yahoo zu integrieren und (ähnlich wie Mailvelope) den Klartext in die Webseite eingebunden darzustellen.

Auch die Erweiterung von Yahoo befindet sich zum Zeitpunkt der Untersuchung in einer Entwicklungsversion, welche manuell kompiliert und installiert werden muss. Die Integration der Erweiterung in die Webseite von Yahoo konnte nicht getestet werden.

6 Bewertung vorhandener Browser-Erweiterungen

Im Folgenden wird dargestellt, inwieweit die Anforderungen aus Kapitel 2 durch die im Kapitel 5 vorgestellten Browser-Erweiterungen erfüllt werden. Die Nummerierung der Anforderung wird aus Abschnitt 2.8 übernommen.

Erweiterung	**A1** Klartextab-sicherung	**A2** Key Discovery & Authentification	**A3** Einfache Installation	**A4** Integration zu GnuPG	**A5** Universelle Einsetzbarkeit	**A6** PGP/MIME-Unterstützung	**A7** Codequalität
Mailvelope	Ja	Nein	Ja	Nein	Ja	Ja	Ausreichend
PGP Anywhere	Ja	Nein	Ja	Nein	Nein	Nein	Nein
Mymail-Crypt for Gmail	Nein	Nein	Ja	Nein	Nein	Nein	Nein
Secure Email	Nein	Ja	Nein	Ja	Nein	Nein	Nein
Enlocked	Nein	Nein	Ja	Nein	Nein	Nein	Nein
WebPG	Nein	Ja	Nein	Ja	Ja	Nein	Nein
End-To-End	Ja	Unbekannt	Ja	Nein	Nein	Unbekannt	Ja

6.1 Absicherung des Klartextes gegenüber der Webanwendung (A1)

Die Anforderung den zu verschlüsselnden Klartext gegen die Webseite zu schützen, in welche die Nachricht eingegeben wird, ist vermutlich der größte Aufwand bei der Entwicklung einer Erweiterung. Während für die kryptografischen Funktionen auf fertige Bibliotheken wie OpenPGP.js zurückgegriffen werden kann, muss diese Anforderung durch die Entwickler der Erweiterung vollständig selbst vorgenommen werden.

Von den vorgestellten Erweiterungen wird diese Anforderung nur von Mailvelope, PGP Anywhere und End-To-End erfüllt. Bei allen anderen Erweiterungen wird der Klartext in ein Textfeld der Webseite eingegeben, so dass diese auf den Klartext zugreifen kann.

PGP Anywhere löst das Problem, in dem die Erweiterung überhaupt nicht mit der Webseite interagiert. Dieses Vorgehen bietet zwar den besten Schutz, jedoch stellt sich die Frage, wieso die Funktionalität als Browser-Erweiterung angeboten wird. Genauso gut könnte hierfür eine browserunabhängige Desktop-Anwendung verwendet werden.

Das gleiche gilt auch für End-To-End von Google – jedenfalls im aktuellen Entwicklungsstand. Auch hier wird der Klartext nur dadurch geschützt, dass dieser nicht in die Webseite integriert wird. Bei der Implementierung von End-To-End von Yahoo soll die Absicherung über ein iFrame realisiert werden, welches von der Erweiterung zur Verfügung gestellt wird und den Klartext der verschlüsselten Daten von dem Mailprovider schützt. Inwiefern dies in der aktuellen Entwicklungsversion funktioniert, konnte nicht getestet werden.

Mailvelope bietet einen Editor an, welcher sich in die besuchte Webseite integriert. Der Gefahr von Phishing-Angriffen wird durch die individuelle Gestaltung des Editors begegnet. Die Integration gelingt besonders gut

auf Webseiten, welche für Mailvelope optimiert sind – der Ansatz funktioniert jedoch auf beliebigen Seiten, auf welchen der Nutzer Text eingeben kann.

6.2 Einfacher und sicherer Schlüsselaustausch (A2)

Soweit die Erweiterung auf GnuPG als OpenPGP-Implementierung basiert und somit auch die Schlüsselverwaltung von GnuPG verwendet wird, stehen auch die Funktionen der Key Discovery und Authentification von GnuPG zur Verfügung. Bei den untersuchten Anwendungen ist dies bei WebPG und Secure Email der Fall.

Die untersuchten Erweiterungen, die auf OpenPGP.js aufbauenden, bieten solche Funktionen nicht an. Auch von Mailvelope werden diese Funktionen nicht angeboten. Die Erweiterung geht viel mehr davon aus, dass allen in den Schlüsselbund aufgenommenen öffentlichen Schlüsseln vertraut wird. Hierbei können öffentliche Schlüssel nur manuell in den Schlüsselbund importiert werden.

Die End-To-End-Erweiterungen von Google und Yahoo bieten Funktionen zur Key Discovery an, in dem sich die Erweiterung mit speziellen Keyservern verbindet. Inwieweit dies zum aktuellen Zeitpunkt funktioniert, konnte nicht getestet werden.

6.3 Einfache Installation (A3)

Wie in Abschnitt 2.3 dargestellt, ist die Installation der Erweiterung immer dann einfach, wenn nur die Erweiterung installiert werden muss und keine weiteren Programme (wie GnuPG) vorausgesetzt werden. Dies wird von allen Erweiterungen erfüllt, die auf einer JavaScript-Bibliothek (wie OpenPGP.js) basieren. Also alle, bis auf WebPG und Secure Email.

Die End-To-End-Erweiterungen von Google und Yahoo können zum aktuellen Zeitpunkt nur sehr aufwendig installiert werden. Sie müssen manuell kompiliert und über eine Entwickleroption in Chrome aktiviert werden. Da die Erweiterungen jedoch nur auf Javascript basieren, wird es nach ihrer Fertigstellung möglich sein, sie einfach über den Chrome Web Store zu installieren.

6.4 Integration mit GnuPG (A4)

Die Anforderung, sich in eine bestehende GnuPG-Installation zu integrieren, wird aktuell nur von den Erweiterungen erfüllt, welche auf GnuPG basieren – konkret WebPG und Secure Email.

Die Anforderung scheint im Widerspruch zu einer einfachen Installation zu stehen. Wie im Fazit im Abschnitt 3.3 beschrieben, ist es möglich, dass eine Erweiterung zwar auf GnuPG basiert, jedoch auf eine JavaScript-Bibliothek wie OpenPGP.js zurückgreift, wenn GnuPG nicht installiert ist. Auf diese Weise können beide Anforderungen erfüllt werden. Aktuell ist dies jedoch bei keiner der untersuchten Erweiterungen der Fall.

6.5 Universelle Einsetzbarkeit (A5)

Von den untersuchten Erweiterungen sind lediglich Mailvelope und WebPG universell einsetzbar.

Die Erweiterungen PGP-Anywhere, gmail-crypt, Enlocked und End-To-End stehen nur für den Browser Chrome zu Verfügung. Mit den Erweiterungen gmail-crypt und Enlocked lassen sich nur E-Mails über das Portal gmail.com versenden.

Die Erweiterung Secure Email funktioniert derzeit nur unter Windows und nur für Outlook Web App.

Bei Secure Email und Enlocked handelt es sich um einen geschlossenen Service, bei welchem Nutzer nur mit anderen Nutzern des selben Dienstes verschlüsselt kommunizieren können.

Lediglich Mailvelope und WebPG sind unter Windows, Mac OS und Linux auf Firefox und Chrome einsetzbar, wobei auch eine Kompatibilität zu anderen OpenPGP-Implementierungen besteht. Daher, dass auch Texte, die mit GnuPG verschlüsselt wurden, wieder entschlüsselt werden können.

6.6 Unterstützung von PGP/MIME (A6)

Wie in Anforderung A6 im Abschnitt 2.6 dargelegt, ist es für eine Unterstützung von PGP/MIME erforderlich, dass die Erweiterung mit dem Mailprovider über eine API kommuniziert. Von den untersuchten Erweiterungen wird eine solche API nur von Mailvelope angeboten, wodurch nur Mailvelope in der Lage ist PGP/MIME-E-Mails zu versenden und zu empfangen - jedoch auch nur dann, wenn der Mailprovider die API unterstützt.

Da sich die Erweiterung End-To-End von Yahoo in deren Webmailer integriert, kann davon ausgegangen werden, dass die Erweiterung auch PGP/MIME unterstützten wird. Zum aktuellen Zeitpunkt konnte dies jedoch nicht getestet werden.

6.7 Entwicklungsprozess, Entwicklergemeinschaft und Codequalität (A7)

Bei der Codequalität der untersuchten Erweiterungen muss zwischen der Erweiterung selbst und der verwendeten OpenPGP-Implementierung unterschieden werden.

Die drei verwendeten OpenPGP-Implementierungen erfüllen die aufgestellten Anforderungen. Bei allen handelt es sich um Freie Software. Alle haben eine ausreichende Entwicklergemeinschaft, welche aktiv an der jeweiligen Bibliothek entwickelt. Alle drei Bibliotheken werden durch Unit-Tests getestet und wurden bereits unabhängigen Security Audits unterzogen.

Der Entwicklungsprozess von Mailvelope findet offen auf GitHub statt. Die Software wird aktiv entwickelt und wurde verschiedenen Security Audits unterzogen, welche teilweise veröffentlicht wurden. Unit-Tests sind für Mailvelope nicht vorhanden, jedoch gibt es eine Integration-Testsuite, welche speziell die API von Mailvelope testet. Insbesondere in Bezug auf die Unit-Tests kann die Codequalität von Mailvelope daher noch verbessert werden.

PGP Anywhere wird zwar offen auf GitHub entwickelt, eine Entwicklergemeinschaft ist jedoch nicht vorhanden. Der öffentliche Bug Tracker listet weder offene noch geschlossene Bugs. Die Software kann auch keine automatischen Tests oder öffentliche Security Audits vorweisen. Die Codequalität ist somit für eine sicherheitsrelevante Anwendung nicht ausreichend.

Das GitHub-Profil von Mymail-Crypt for Gmail kann zwar eine gewisse Aktivität vorweisen, jedoch ist auch diese insgesamt gering. Lediglich ca. 70 Issues wurden seit der Einrichtung auf GitHub eingereicht. Automatische Tests und öffentliche Security Audits sind nicht vorhanden. Die Codequalität ist daher ebenfalls nicht ausreichend.

WebPG wird aktuell ebenfalls nicht aktiv entwickelt. Die letzten Commits für Firefox sind von Juni 2014, die letzten Commits zu Chrome von September 2015. Auch ansonsten ist keine Aktivität ersichtlich. Öffentliche Security Audits werden nicht aufgelistet. Die Codebasis beinhaltet jedoch sowohl für Firefox als auch für Chrome automatische Tests.

Da es sich bei Secure Email und Enlocked um proprietär Software handelt, kann eine positive Aussage hinsichtlich der Codequalität nicht getroffen werden. Auf der Internetseite des jeweiligen Herstellers konnten keine Informationen über öffentliche und unabhängige Security Audits gefunden werden.

Die Entwicklung von End-To-End ist sowohl bei Google als auch bei Yahoo sehr aktiv. In beiden Repositories flossen in den letzten Monaten viele Commits ein. Die Software ist gut getestet.

7 Im Browser integrierte Verschlüsselung

Wie zu Beginn der Studie erläutert, wäre es vorstellbar, dass die verschiedenen Browser die Ende-zu-Ende-Verschlüsselung direkt unterstützen und somit keine Erweiterung installiert werden müsste. Bei einer solchen Implementierung könnten die Entwickler einer Webseite auf HTML-PGP-Tags zurückgreifen, durch welche der Nutzer Texte ent- und verschlüsseln kann.

In diesem Abschnitt sollen Gesichtspunkte erläutert werden, die bei einer Entwicklung von HTML-PGP-Tags berücksichtigt werden müssten.

7.1 Eingabe von Daten

7.1.1 Eingabe verschlüsselter Texte

Die Eingabe von Texten im Webbrowser erfolgt üblicherweise über die Tags `<input>` und `<textarea>`. Bei beiden Tags kann JavaScript, welches von der Webanwendung geladen wurde, auf den eingegebenen Text zugreifen, da der Inhalt Bestandteil der zugänglichen DOM-Datenstruktur der Webseite ist. Beispielsweise ist es möglich, dass sich der Quellcode der Webseite über das onchange-Event über jede Änderung in einem bestimmten Tag informieren lässt und diese Änderung an den Server sendet. Auch ohne JavaScript wird der Inhalt der Textfelder an den Server gesandt, sobald der Nutzer auf einen Submit-Button innerhalb des Formulars drückt.

Aus diesem Grund sind die Tags in ihrer aktuellen Form ungeeignet Texte entgegenzunehmen, welche Ende-zu-Ende-verschlüsselt werden sollen. Der Nutzer kann nicht darauf vertrauen, dass der Webseite der Klartext der E-Mail verborgen bleibt.

Der HTML-Standard müsste daher um eine Möglichkeit ergänzt werden Text einzugeben, auf den die Webanwendung keinen Zugriff hat – weder per JavaScript, noch über das Versenden des Formulars per Submit-Button. Stattdessen müsste vor dem Auslesen und Senden des Formularinhalts der Text verschlüsselt werden.

Weiterhin muss das Formularelement sich auf eine Art und Weise von normalen Elementen abgrenzen, so dass der Nutzer auf den ersten Blick erkennen kann, ob der eingegebene Text verschlüsselt wird oder nicht. Es ist insbesondere darauf zu achten, dass ein normales Webseiten-Element nicht mittels CSS so gestaltet werden kann, dass es wie ein PGP-Eingabefeld aussieht.

Das Problem könnte in der gleichen Art und Weise gelöst werden, wie es die bereits vorgestellte Browser-Erweiterung Mailvelope löst. Nach dieser Lösung könnte der Hintergrund der PGP-Tags durch den Nutzer individuell gestaltet werden.

Um Daten zu verschlüsseln, muss der Browser wissen, mit welchen öffentlichen Schlüsseln der Text verschlüsselt werden soll. Hierfür ist es komfortabel, wenn diese Zertifikate mit Wissen des Webmailers ausgewählt werden können. Beim Versenden einer E-Mail wäre es für die Auswahl der Zertifikate sinnvoll, wenn der Browser die Empfangsadressen kennt. Er wären dann in der Lage passende Zertifikate vorzuschlagen. Dies wäre über ein weiteres HTML-PGP-Tag möglich.

7.1.2 Eingabe signierter Texte

Soweit ein Text nur signiert und nicht verschlüsselt werden soll, spielt es keine Rolle, ob die Webanwendung auf den Klartext zugreifen kann. Wichtig ist jedoch, dass die Webanwendung keinen weiteren Text in das Textfeld einfügen kann. Anderenfalls könnte kurz vor dem Signaturvorgang der Text verändert und danach sofort zurück geändert werden. Hierdurch würde durch den Nutzer ein Text signiert, welcher nicht signiert werden sollte.

Ein weiteres Problem besteht darin, dass bei einem signierten Text zwischen dem Text und der Signatur unterschieden werden muss. Dies wird bei E-Mails im PGP/INLINE-Format dadurch gelöst, dass der Beginn der Signatur gesondert durch Textmarkierungen gekennzeichnet wird.

Beispielsweise:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hier steht irgend ein
von GnuPG signierter Text
[...]
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.2.1 (GNU/Linux)
Comment: For info see http://www.gnupg.org

iD8DBQE5Pf40WyoKbftXl6kRAsWJAJ4hj7FzPX8M9MWZav9u6yjbHXWGKwCfSiKA
wTaJ/lfY1ETv3R/uJrtGTbI=
=BDOH
-----END PGP SIGNATURE-----
```

Dieser Textblock sieht unverständlich aus und kann technisch weniger versierte Benutzer verwirren. Aus diesem Grund könnte ein HTML-PGP-Tag zwischen dem Text und der Signatur unterscheiden.

Dies könnte beispielsweise dadurch realisiert werden, dass ein solcher Tag automatisch ein weiteres hidden-input Feld anlegt, in welches die Signatur eingetragen wird.

7.1.3 Eingabe von verschlüsselten Dateien (Anhänge)

In einem Webformular werden Dateianhänge über ein `<input>`-Tag mit dem Attribute *type=file* realisiert. Für dieses gilt das Gleiche wie für die Eingabe von verschlüsseltem Text. Auch hier muss garantiert sein, dass die Webanwendung keinen Zugriff auf den Klartext der Datei erhält. Weiterhin muss sich der `<input>`-Tag von normalen Upload-Feldern abheben, so dass Angriffe durch Imitation verhindert werden.

7.1.4 Besonderheiten bei E-Mail

Die Problematik beim Versenden von E-Mails ist bei einer Implementierung im Browser und der Implementierung in einer Erweiterung identisch. E-Mails können ohne weiteres als PGP/INLINE versandt werden. Für das Versenden einer E-Mail im bevorzugten PGP/MIME-Format ist es jedoch erforderlich, dass der PGP/MIME-Block durch den Browser erstellt wird. Hierfür müsste der Browser eine entsprechende JavaScript-Bibliothek anbieten.

7.2 Anzeige von Daten

7.2.1 Anzeige von verschlüsselten Texten

Für das Anzeigen von verschlüsseltem Text könnte ein Inline-Tag geschaffen werden. Der Browser kann den Inhalt dieses Tags automatisch entschlüsseln und den entschlüsselten Text anzeigen. Auch an dieser Stelle muss darauf geachtet werden, dass die Webanwendung nur auf den verschlüsselten Text zugreifen kann und keine Imitation der Anzeige durch den Server möglich ist. Dies bedeutet, dass zwar der entschlüsselte Text für den Nutzer angezeigt wird, über den DOM jedoch nur der verschlüsselte Text aufgerufen werden kann.

7.2.2 Anzeige von signierten Texten

Beim Anzeigen von signiertem Text besteht die Herausforderung darin, dass das Ergebnis der Validierung auf eine Art und Weise angezeigt wird, welche die Webseite nicht fälschen kann.

Soweit der Text nur signiert und nicht verschlüsselt ist, sollte der Tag zwischen dem signierten Text und der Signatur unterscheiden. Auf diese Weise könnte die Signatur nur dann ausgewertet werden, wenn der Browser das Tag unterstützt. Ältere Browser würden in diesem Fall nur den Text ohne die Signatur anzeigen. Es könnte sich daher in etwa so verhalten, wie die folgenden HTML-Tags:

```
<div class="signed-content">
  <span>Dieser Text ist signiert.</span>
  <span style="display:hidden;">SIGNATUR</span>
</div>
```

7.2.3 Anzeigen von verschlüsselten Dateien (Anhänge)

Es muss darauf geachtet werden, dass die Webanwendung keinen Zugriff auf die entschlüsselten Dateien erhält. Ein Tag oder ein Attribut könnte anzeigen, welche Anhänge vermutlich verschlüsselt sind, damit der Browser diese der PGP-Erweiterung übermitteln kann. Dort kann sie lokal (außerhalb des Browsers) angezeigt oder gespeichert werden.

7.3 Anbindung an GnuPG

Die Definition des PGP-Tags ist unabhängig von der darunter liegenden OpenPGP-Implementierung. Theoretisch könnte der Browser alle Funktionen nach RFC 4880 selbst implementieren. Soweit der Browser auf eine bestehende Lösung (wie beispielsweise GnuPG) zurückgreift, besteht der Vorteil, dass die Nutzer seine bestehende Schlüsselverwaltung verwenden kann. Die privaten und öffentlichen Schlüssel können in diesem Fall von GnuPG erstellt und gepflegt werden. Soweit der Browser die Funktionalität selbst implementiert, bedeutet dies, dass er auch eine eigene Schlüsselverwaltung implementieren muss und die Schlüssel aus GnuPG nicht verwendet werden können. Aus diesem Grund sollte der Browser eine Schnittstelle schaffen, über welche die kryptografischen Operationen an eine lokale Anwendung wie GnuPG weitergereicht werden können.

7.4 Realisierbarkeit

Um entsprechende PGP-Tags einzuführen, müsste der HTML-Standard erweitert werden. Dies ist ein aufwendiges Verfahren, bei welchem eine Vielzahl von Personen beteiligt ist. Die tatsächliche Umsetzung der PGP-Tags erscheint den Autoren der Studie daher als unwahrscheinlich.

8 Risko- und Aufwandsabschätzung zur Weiterentwicklung

Anhand der aufgestellten Anforderungen zeigt sich, dass Mailvelope von den untersuchten Erweiterungen das beste Potenzial aufweist und sich daher für eine Weiterentwicklung anbietet.

Der Hauptgrund hierfür liegt darin, dass Mailvelope als einzige der untersuchten Erweiterungen in der Lage ist, mit verschiedenen Mailprovidern zu kommunizieren und E-Mails im PGP/MIME-Format zu versenden. Weiterhin ist nur Mailvelope in der Lage, sich zum einen in den Webmailer zu integrieren, jedoch gleichzeitig eine sichere Umgebung zu schaffen, in welcher der Klartext der Daten auch vor dem Webmailer geschützt ist. Diese Punkte stellen die Hauptschwierigkeit bei der Entwicklung einer Ende-zu-Ende-Erweiterung dar.

Von den aufgestellten Anforderungen fehlt Mailvelope das Key Discovery und Key Authentication (A2) und eine Anbindung an GnuPG (A4). Weiterhin kann die Codequalität von Mailvelope dadurch verbessert werden, dass die Funktionalität durch Unit-Tests automatisch getestet wird.

Die Alternative hierzu würde darin bestehen, auf eine Erweiterung aufzusetzen, welche bereits auf GnuPG basiert, und diese um einen sicheren Editor zu erweitern. Von den untersuchten Erweiterungen kommt hierfür WebPG in Betracht. Aufgrund dessen, dass WebPG jedoch in der aktuellen Version noch auf NPAPI basiert und die Entwicklungsaktivitäten gering sind, bietet selbst die Anbindung an GnuPG zur Zeit keine ausreichende Grundlage. Selbst wenn jedoch WebPG in einer zukünftigen Version auf Native Messaging basieren sollte, müsste neben dem sicheren Editor auch eine API implementiert werden, über welche E-Mails im PGP/MIME-Format versandt werden können. Aus diesen Gründen ist nach der Einschätzung der Autoren die Weiterentwicklung von Mailvelope zielführender.

Es werden die folgenden vier Arbeitspakete vorgeschlagen, um Mailvelope an die fehlenden Anforderungen anzupassen. In der Summe ergibt sich daraus ein ungefährer Entwicklungsaufwand in Höhe von 210 Personentagen.

Danach folgen Weiterentwicklungswege für OpenPGP.js mit grober Abschätzung.

Die anhand von Mailvelope für E-Mail gezeigten Möglichkeiten für Ende-zu-Ende Kryptografie in Webanwendungen sind für eine Reihe von weiteren Anwendungsfeldern denkbar, besonders wenn von der Seite der Web-Browser und Server mehr Unterstützungs-Funktionalität implementiert werden kann. Denkbar sind beispielsweise Authentifikationsfunktionen für Darstellungselemente über Signaturen oder eine integrierte Handhabung von Dateitransfer. Die Grundbedingungen dafür, dass eine Anwendung in Frage kommt sind: Eine Sicherheitsleistung von OpenPGP wird benötigt und die Daten müssen so einfach strukturiert sein, dass sie lokal auf dem Rechner bearbeitet werden können. Das gilt beispielsweise für Textnachrichten und Anhänge; und trifft nicht auf eine komplexe Webeditor-Anwendung wie eine Tabellenkalkulation zu. Die lohnenswerten Anwendungen konkret auszuarbeiten, geht über das Ziel dieser Studie hinaus. Eine Weiterentwicklung in dieser Richtung müsste einen explorativen Projektcharakter haben und mit Freie Software-Prototypen arbeiten.

8.1 Integration mit GnuPG (A4)

Der wichtigste und gleichzeitig aufwendigste Punkt ist die Anbindung an GnuPG. Wie im Abschnitt 3.3 aufgeführt, gibt es dabei zwei Herangehensweisen:

1. Optionale Anbindung von GnuPG
2. Vollständiger Ersatz der bisherigen Krypto-Funktionen durch GnuPG

Bei beiden Anforderung müssen im ersten Schritt alle Funktionsaufrufe aus OpenPGP.js durch generische abstrakte Funktionen ersetzt werden, welche sowohl eine entsprechende Funktion von GnuPG oder OpenPGP.js aufrufen kann. Im zweiten Schritt muss dann die Kommunikation zwischen Mailvelope und

GnuPG über Native Messaging hergestellt werden. Hierfür ist eine enge Zusammenarbeit mit den Entwicklern von GnuPG erforderlich, um die Schnittstellen von GnuPG entsprechend anzupassen.

Sofern ausschließlich GnuPG für Krypto-Funktion verwendet werden soll, müssten doppelte Funktionen von OpenPGP.js aus Mailvelope entfernt werden. Gleichzeitig muss mehr Aufwand in die Bereitstellung einer leichten Installation von GnuPG gesteckt werden.

Der Aufwand umfasst in etwa 100 Personentage.

8.2 Einfacher und sicherer Schlüsselaustausch (A2)

Soweit die Anforderung A4 erfüllt ist, kann Mailvelope auf die Schlüsselverwaltung von GnuPG zugreifen. Somit wäre bei der Verwendung von GnuPG automatisch auch die Anforderung des Key Discovery und Key Authentication (A2) erfüllt. Um diese Anforderung jedoch auch dann zu erfüllen, wenn GnuPG nicht installiert ist, sind auch hierfür Anpassungen notwendig. Wie in Abschnitt 2.2 aufgeführt, ist es empfehlenswert, wenn Mailvelope hierfür in der Lage ist, auf Keyserver zuzugreifen. Für die Key Authentication bietet sich eine Implementierung von TOFU an, wie es aktuell auch für GnuPG 2.2 entwickelt wird. Zumindest muss Mailvelope um Funktionen erweitert werden, um fremde öffentliche Schlüssel manuell zu überprüfen und zu signieren.

Der Aufwand umfasst in etwa 30 Personentage.

8.3 Entwicklungsprozess, Entwicklergemeinschaft und Codequalität (A7)

Als drittes Arbeitspaket lohnt es sich, die Codequalität von Mailvelope zu verbessern. Konkret sollte Mailvelope um automatische Tests ergänzt werden, welche die Funktionalität der Software sicherstellen.

Der Aufwand umfasst in etwa 20 Personentage.

8.4 Security Audit

Bei größeren Änderungen an Mailvelope, wie der oben genannten Integration von GnuPG, ist ein unabhängiges Security Audit sinnvoll.

Der Aufwand umfasst in etwa 60 Personentage.

8.5 Vervollständigung von OpenPGP.js

Sofern OpenPGP.js eine Kernkomponente von Mailvelope und anderen Krypto-Komponenten bleibt, sollte diese mittelfristig an die Qualität von GnuPG herangeführt und gepflegt werden. Der erste Schritt dazu ist die Vollständigkeit und Kompatibilität mit GnuPG und der OpenPGP-Spezifikation gründlicher zu prüfen und fehlende Funktionen zu identifizieren und deren Umsetzung abzuschätzen. Zum Zeitpunkt der Betrachtung fehlen OpenPGP.js beispielsweise elliptische Kurven, insbesondere die Brainpool-Kurven, welche vom BSI als vorteilhaft angesehen werden. Die für den Zertifikatsaustausch für die Weiterentwicklung von Mailvelope oben beschriebenen Funktionen für A2 lassen sich vermutlich auch allgemeiner auf der Ebene von OpenPGP.js implementieren. Sobald neue Methoden der Zertifikats-Verteilung und -Validierung als Ergebnis des EasyGpg-Projektes vorliegen, ist deren Aufnahme in OpenPGP.js ebenfalls sinnvoll. Für der Zukunftsfähigkeit der Pflege von OpenPGP.js erscheint es sinnvoll im Detail abzuwägen, ob die Software-Entwicklungsmethoden der JavaScript-Bibliotek von End-To-End mittelfristig Vorteile bieten werden.

Der Aufwand dieses Weiterentwicklungspunktes hängt von der Prüfung, den bis dahin bereits von Dritten entwickelten Funktionen und dem gewünschten Umfang der Verbesserungen ab und wird grob innerhalb von 20 bis 200 Tagen sinnvoll sein.