E-Mail- und Datei-Verschlüsselung unter iOS mit S/MIME und PGP

Peter Tondl

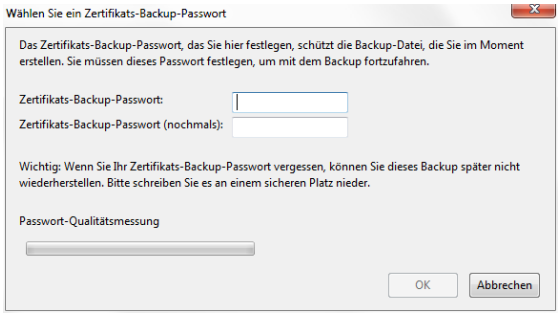
E-Mail- und Datei-Verschlüsselung sind in der Desktop-Welt schon lange kein Thema mehr. Entweder man macht es oder eben auch nicht. An der Technik scheitert es jedenfalls nur selten. Alle relevanten Mail-Programme unterstützen von Haus aus die Verschlüsselung mit S/MIME. Für die Verschlüsselung mit PGP gibt es frei erhältliche Software wie *Enigmail* [1] oder *Gpg4win* [2]. Letzteres stellt nicht nur ein Plugin für Outlook bereit, sondern ermöglicht auch die bequeme Verschlüsselung von Dateien über das Kontextmenü des Datei-Explorers. In der mobilen Welt sieht die Sache etwas anders aus. Sowohl bei der Benutzung von S/MIME als auch bei PGP gibt es ein paar Hürden zu überwinden und Fallen zu vermeiden. Wie das alles geht erläutert am Beispiel von iOS der folgende Bericht.

1 E-Mail und S/MIME

Zuerst die gute Nachricht: Apples E-Mail-Programm für iOS unterstützt von sich aus bereits den Verschlüsselungsstandard S/MIME, so dass keine weiteren Apps benötigt werden.

1.1 Voraussetzungen

Um eine verschlüsselte E-Mail erstellen zu können, benötigt der Sender zunächst einmal ein eigenes X.509-Zertifikat. X.509 ist dabei der Name für den Standard, nachdem solche Zertifikate aufgebaut sind. Eine Zerfifizierungsstelle bestätigt damit in kryptografisch abgesicherter Form, dass beispielsweise die E-Mail-Adresse eines Zertifikatsantragstellers zu einem eingereichten öffentlichen Schlüssel gehört. Zusammen mit dem privaten Schlüssel, den außer der Besitzer niemand kennt und kennen darf, sind auf Seiten des Senders alle Voraussetzung gegeben, um verschlüsselt E-Mails austauschen zu können. Wie das für die sichere Verschlüsselung notwendige Schlüsselpaar, bestehend aus einem öffentlichen Schlüssel und einem geheimen privaten Schlüssel erzeugt werden kann und wie der öffentliche Schlüssel bei einer Zertifizierungsstelle eingereicht wird um zusammen mit der Signatur dieser Zertifizierungsstelle zu einem vollwertigen X.509-Zertifikat zu werden, wird im Anhang beschrieben.



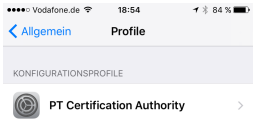
Zertifikatexport aus Firefox

1.2 Eigenes Schlüsselpaar sicher auf das iOS-Gerät bekommen

Ist der Prozess zur Erzeugung des persönlichen X.509-Zertifikats erfolgreich durchlaufen worden, befindet sich das Zertifikat zusammen mit dem erzeugten Schlüsselpaar typischerweise im Sicherheitsmodul des verwendeten Internet-Browsers.

Aus diesem muss es zunächst für die Verwendung mit iOS exportiert werden. Wie dies genau funktioniert ist vom jeweiligen Browser abhängig, das Prinzip ist aber immer das Gleiche. Wurde Mozillas Firefox verwendet, läuft der Pfad zum eigenen Zertifikat beispielsweise über *Extras* → *Einstellungen* → *Erweitert* → *Zertifikate* → *Zertifikate anzeigen* → *Ihre Zertifikate*. Hier angekommen kann das Zertifikat über *Sichern...* in eine PKCS12-Datei (*.p12) exportiert werden.

Zertifikatimport in den Profile-Bereich



Zertifikat einer Zertifizierungsstelle

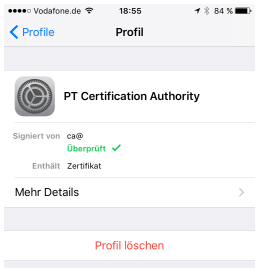
Für den Export sollte unbedingt ein gutes Passwort zum Einsatz kommen, da der private Schlüssel damit gegen fremden Zugriff geschützt wird. Diese Passwort wird nach dem Import des Zertifikats in iOS nicht mehr benötigt, also darf es ruhig etwas länger und komplexer sein als üblich. Obwohl sich nach der reinen Lehre eine Übertragung des *eigenen* Zertifikats – und damit auch des privaten Schlüssels – über E-Mail verbietet, wird dies doch immer wieder gern gemacht. Grund dafür ist sicherlich der umständliche Umgang mit Dateien unter iOS, bei dem E-Mail oftmals der einzige Weg zum Austausch von Daten zwischen Umwelt und iOS-Gerät ist. Um das Risiko auf dem Übertragungsweg wenigstens zu minimieren empfiehlt es sich, das exportierte Zertifikat an sich selbst zu senden, damit die E-Mail (hoffentlich) den Server des Anbieters nicht verlässt. Ist der Übertragungsweg von und zum E-Mailserver wie heute eigentlich üblich über SSL gesichert, kann auch auf diesem Weg das Abfangen der E-Mail durch Dritte verhindert werden. Nach dem Import des Zertifikats sollte dann unbedingt noch die E-Mail gelöscht werden und zwar im Ordner für die empfangenen Mails *und* im Ordner für die gesendeten Mails.

1.3 E-Mail-Account für die Verwendung mit S/MIME einrichten

Vertrauenseinstellung für Zertifizierungsstellen

Der Import des eigenen Zertifikats erfolgt durch Antippen des E-Mail-Anhangs. iOS fordert den Benutzer zur Eingabe des oben beim Export festgelegten Passworts auf. Nach erfolgreichem Eintippen wird das Zertifikat inklusive des privaten Schlüssels in iOS importiert und ist fortan im Profile-Bereich unter *Einstellungen* → *Allgemein* → *Profile* zu sehen. In diesem Bereich werden auch die sogenannten Wurzel- oder Root-Zertifikate der Zertifizierungsstellen abgelegt, die über den oben beschriebenen Weg ebenfalls importiert werden können. Dies ist immer dann nötig, wenn iOS die Zertifizierungsstelle von sich aus nicht kennt, typischerweise bei selbst erstellten Zertifikaten über eine eigene Zertifizierungsstelle, wie es häufig bei größeren Unternehmen der Fall sein kann, die eine eigene Zertifizierungs-Infrastruktur betreiben.

Zertifizierungsstellen mit vollem Vertrauen des Benutzers

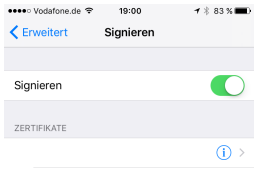


Achtung Falle! Seit iOS 10.3 reicht es nicht mehr wie in früheren Versionen aus, das Zertifikat einer Zertifizierungsstelle nur zu importieren. Von einer solchen Zertifizierungsstelle beglaubigte Zertifikate werden erst dann als gültig eingestuft, wenn dem Wurzel-Zertifikat zusätzlich in den *Einstellungen* → *Allgemein* → *Info* → *Zertifikatsvertrauenseinstellungen* „volles Vertrauen“ zugestanden wurde. Ist vom Benutzer die Einstellung für das Ver-

trauen in die Zertifikate aktiviert worden, werden vom Wurzel-Zertifikat abgeleitete Zertifikate automatisch als gültig angesehen, da sie von diesem ja beglaubigt wurden. Separate Vertrauenseinstellungen für untergeordnete Zertifikate sind damit auch nicht extra erforderlich.

Auf diese Weise manuell hinzugefügte Zertifikate können über den Befehl *Profil löschen* jederzeit wieder aus dem Zertifikatsspeicher entfernt werden. Wird ein Wurzel-Zertifikat entfernt, verlieren damit auch alle davon abhängigen Zertifikate ihre Gültigkeit und können für die Verschlüsselung von neuen E-Mails nicht mehr verwendet werden. Das Entschlüsseln bereits vorhandener E-Mails ist aber nach wie vor möglich, auch wenn dabei eine Warnung wegen der nicht vorhandenen Gültigkeit angezeigt wird.

S/MIME aktivieren



Standardmäßig signieren

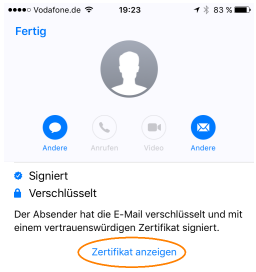
Standardmäßig verschlüsseln

S/MIME für den E-Mail-Account zu aktivieren genügt jedoch noch nicht ganz. Sollen E-Mails auch beim Senden signiert und eventuell verschlüsselt werden, muss zusätzlich noch die Signatur oder Verschlüsselung für ausgehende Mails unter *Einstellungen* → *Mail* (bis iOS 10) *Accounts & Passwörter* (ab iOS 11) → *Accounts* → „Name des E-Mail-Kontos“ → *Account* → *Erweitert* aktiviert werden. Leider lässt sich dies nicht individuell für jede E-Mail einstellen – zwar kann die Verschlüsselung für eine bestimmte E-Mail ausgeschaltet werden, wenn sie für das E-Mail-Konto zuvor generell eingeschaltet wurde. Der umgekehrte Fall ist jedoch nicht möglich. Auch lässt sich das Signieren nicht abschalten, wenn es für das Konto eingeschaltet wurde. Signiert wird in diesem Fall also immer.

Standardmäßig signieren genauso wie *Standardmäßig verschlüsseln* ist nur möglich, wenn ein eigenes, gültiges Zertifikat vorliegt. Alle vorhandenen eigenen und gültigen Zertifikate werden in den jeweiligen Einstellungsseiten zur Auswahl angeboten. Ist eine Zuordnung eindeutig möglich, wählt iOS von sich aus auch schon ein Zertifkat aus, welches dann mit einem Häckchen gekennzeichnet ist.

1.4 E-Mails mit S/MIME signieren und verschlüsseln

Verschlüsselte und signierte E-Mail empfangen



Empfangenes Zertifikat anzeigen lassen

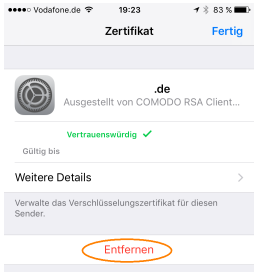
Empfangenes Zertifikat installieren

Sind alle Zertifikate an Bord und alle Schalter gesetzt kann es endlich losgehen. Zumindest mit dem Signieren eigener Mails. Für das Verschlüsseln fehlt aber immer noch das passende Zertifikat des Kommunikationspartners. Ohne dieses geht es nicht, da das Mail-Programm sonst keinen Schlüssel hat, für den es eine Nachricht verschlüsseln soll. Die einfachste Möglichkeit an einen solchen Schlüssel zu kommen, ist, eine signierte E-Mail vom gewünschten Mail-Partner zu bekommen. Einer mittels S/MIME signierten Mail hängt immer das Zertifikat des Absenders an, welches durch ein paar Fingertipps in den eigenen Zertifikatsspeicher übernommen werden kann. Fortan ist das Verschlüsseln an diesen E-Mail-Partner möglich, solange dessen Zertifikat gültig ist.

Achtung Falle! Zertifikate von Kommunikationspartnern werden nicht im oben beschriebenen Profile-Bereich abgespeichert. Es gibt aber auch sonst keinen Bereich innerhalb von iOS, ob in den Einstellungen oder sonstwo, über den auf

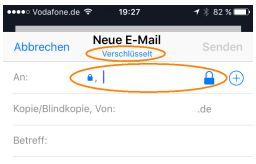
die installierten Zertifikate zugegriffen werden kann. Das führt zu einem unangenehmen Effekt: Soll das Zertifikat zu einer E-Mail-Adresse durch ein neues Zertifikat ersetzt werden, kann dieses zwar scheinbar wie oben beschrieben installiert werden, tatsächlich wird das alte Zertifikat aber eben nicht durch das neue überschrieben. Es besteht somit also keine Möglichkeit, das Zertifikat eines E-Mail-Partners auszutauschen oder auch einfach nur zu entfernen!

IMAP-Ordner für empfangene Zertifikate



Empfangenes Zertifikat entfernen

Glücklicherweise gibt es einen einfachen Trick, um dieser Misere aus dem Weg zu gehen. Werden die Eigenschaften eines einer E-Mail angehängten und bereits im Speicher von iOS vorhandenen Zertifikats aufgerufen, wird von iOS nicht mehr die Möglichkeit zur Installation des Zertifikats angeboten, sondern nunmehr die Möglichkeit zum Entfernen. Es wird also lediglich zu jedem installierten Zertifikat eines Kommunikationspartners eine von diesem stammende, signierte E-Mail benötigt, um dieses Zertifikat wieder aus dem eigenen Speicher entfernen zu können. Um für diesen Zweck immer auf eine passende E-Mail zugreifen zu können, kann beispielsweise im E-Mail-Konto ein eigener Ordner angelegt werden, in dem die E-Mail, die ursprünglich zur Installation benutzt wurde, zum Zweck des späteren Entfernens des zugehörigen Zertifikats abgelegt wird.

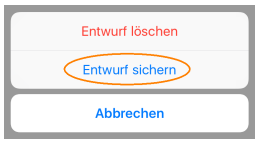


Verschlüsselte E-Mail erstellen

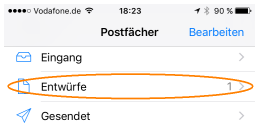
Ist auch diese Hürde genommen, kann es endlich losgehen. Die Mail-App zeigt abhängig vom eingetragenen Empfänger über ein Schlosssymbol an, ob ein Verschlüsseln möglich ist (Signieren ist immer möglich und wird auch immer gemacht, wenn *Standardmäßig signieren* aktiviert ist). Gleichzeitig erscheint bei möglicher Verschlüsselung, das heißt, es ist für jeden (!) Empfänger ein gültiges Zertifikat vorhanden, auch noch der Hinweis *Verschlüsselt* im Kopfbereich der E-Mail. Um die Verschlüsselung für diese E-Mail auszuschalten, genügt ein Tipp auf das große Schlosssymbol, welches sich daraufhin öffnet. Ist ein Verschlüsseln nicht möglich, wird dies im Kopfbereich durch einen in rot erscheinenden Hinweis sehr deutlich kundgetan. Häufigste Ursache dafür dürfte sein, dass das Zertifikat eines E-Mail-Partners entweder noch nicht vorhanden oder nicht mehr gültig ist.

1.5 E-Mail-Verschlüsselung aus anderen Apps heraus

Eigentlich dürfte es diesen Abschnitt gar nicht geben. Aber aus unerfindlichen Gründen lässt sich die E-Mail-Verschlüsselung mittels S/MIME in der Mail-App *nicht* aktivieren, wenn diese aus einer anderen App, beispielsweise über den *Senden an*-Befehl, aufgerufen wurde (Stand mindestens bis iOS 11.1.2).



E-Mail-Entwurf sichern



E-Mail-Entwurf wieder aufrufen

Lokales Speichern für Entwürfe und Papierkorb aktivieren

Achtung Falle! In diesem Fall wird also immer unverschlüsselt und auch unsigniert versendet. Glücklicherweise gibt es aber auch für dieses Problem, bis es denn von Apple in ferner Zukunft vielleicht doch noch einmal behoben wird, zumindest eine alternative Vorgehensweise. Der Trick besteht darin, die E-Mail nach ihrer Erstellung zunächst einmal nicht zu senden, sondern den Vorgang abzubrechen. Die Mail-App fragt daraufhin an, ob der Entwurf gesichert oder gelöscht werden soll. Natürlich ist *Entwurf sichern* hier die richtige Wahl, denn die Mail soll ja ihren Empfänger irgendwann erreichen. Anschließend muss der E-Mail-Entwurf aus dem Verzeichnis für Entwürfe wieder aufgerufen werden. Der Unterschied zum direkten Senden besteht darin, dass jetzt die Ver-

schlüsselung der E-Mail wie durch ein Wunder möglich ist.
Es bleibt noch das Problem zu lösen, dass der unverschlüsselte Entwurf beim Sichern zumindest bei Verwendung von IMAP als E-Mail-Protokoll zunächst auf dem Server des Anbieters dieser Dienstleistung landet. Da dies im Allgemeinen nicht gewollt sein dürfte, müssen die Speicherorte für die Verzeichnisse *Entwürfe* und am besten auch gleich für den *Papierkorb* unter *Einstellungen* → *Mail* (bis iOS 10) *Accounts & Passwörter* (ab iOS 11) → *Accounts* → „Name des E-Mail-Kontos“ → *Account* → *Erweitert* vom E-Mail-Server auf das iOS-Gerät umgebogen werden. Somit verlässt kein unverschlüsselter Entwurf und keine ebenfalls unverschlüsselte und zu löschende E-Mail mehr das Gerät.

2 E-Mail und Datei-Verschlüsselung mit PGP

Während S/MIME bereits in Apples Mail-App fest integriert ist, gilt es für die Verwendung von PGP als Verschlüsselungssystem etwas mehr Aufwand zu treiben. Belohnt wird dieser jedoch mit der zusätzlich gewonnenen Funktionalität, nunmehr auch Dateien unabhängig von ihrem Versand über eine E-Mail kryptografisch sichern zu können, da PGP im Gegensatz zu S/MIME nicht auf den Einsatz mittels E-Mail beschränkt ist.

2.1 Voraussetzungen

Für iOS empfehlen sich zwei kostenpflichtige Apps (derzeit 2,29 € und 5,49 €), die beide – wie so oft – ihre Vor- und Nachteile haben.

iPGMail

oPenGP

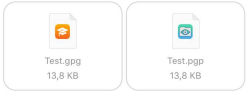
iPGMail nennt als Systemvoraussetzung iOS 8.0 oder neuer [3]. Für ältere Versionen von iOS lässt sich zwar die letzte kompatible Version der App ebenfalls herunterladen, wenn unter Benutzung der gleichen Apple-ID zuvor die aktuelle App-Version auf einem iOS 8-fähigem Gerät installiert wurde. Diese stürzt aber unter iOS 7 sang- und klanglos ab. Die letzte Aktualisierung der App erfolgte im Juli 2017.
oPenGP ist von Anfang an bescheidener, hier genügt als Systemvoraussetzung iOS 7 oder neuer. Dies kann natürlich auch daran liegen, dass die App seit Dezember 2015 nicht mehr aktualisiert wurde. Dennoch handelt es sich um keine reine 32-Bit-App, sie läuft also unter iOS 11 auch noch ohne Probleme.

2.2 Eigenes Schlüsselpaar sicher auf das iOS-Gerät bekommen

Beide Apps unterstützen die Übertragung des privaten PGP-Schlüssels mittels iTunes-Datenaustausch und USB-Kabel. Damit ist eine besonders sichere Übertragung des privaten Schlüssels vom PC auf das iOS-Gerät möglich. Ist die oben schon beschriebene Verschlüsselung von E-Mails mittels S/MIME möglich, kann auch diese Methode gefahrlos genutzt werden. Generell ist aber immer zu bedenken, dass die Verwendung eines besonders „wertvollen“, da unter Umständen seit Jahren benutzten und somit bei vielen Kommunikationspartnern bekannten und akzeptierten PGP-Schlüssels auf einem Mobilgerät sicher nicht ganz unproblematisch ist. Schließlich ist die Gültigkeit eines solchen Schlüssels nicht wie bei S/MIME auf meistens ein Jahr beschränkt. Eine ordentliche Absicherung des iOS-Geräts gegen Fremdbenutzung ist daher Pflicht.

2.3 E-Mail und PGP

Da Apple bei seiner Mail-App keine Erweiterungen erlaubt, ein Austausch des Standard-E-Mail-Programms aber auch nicht zulässt, kann iPGMail, wie auch oPenGP, nur als „Beistell-App“ genutzt werden: Wird eine E-Mail mit PGP empfangen, also entweder eine PGP/MIME-kodierte E-Mail oder eine „normale“ E-Mail mit PGP-verschlüsselter Datei als Anhang, kann dieser Anhang über ein Antippen in der E-Mail an iPGMail oder oPenGP weitergegeben werden.

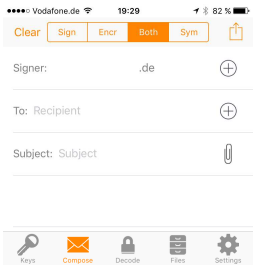


PGP-Anhänge

Achtung Falle! Anhänge für PGP sind in einer E-Mail nicht unbedingt als solche zu er-

kennen. Beispielsweise kann es in Abhängigkeit von der eigenen Gerätekonfiguration durchaus passieren, dass ein Anhang mit der Endung *.gpg* als *iTunes U*-Datei und ein Anhang mit der Endung *.pgp* als *Goodreader*-Datei angezeigt werden kann. Hier hilft im Zweifel nur Ausprobieren oder genaues Hinsehen.

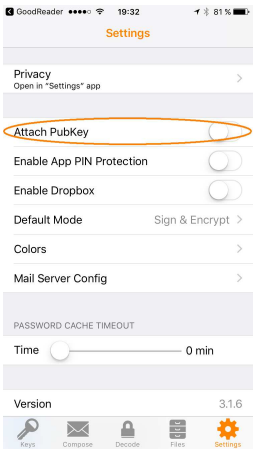
2.4 E-Mails mit iPGMail signieren und verschlüsseln



E-Mail verfassen

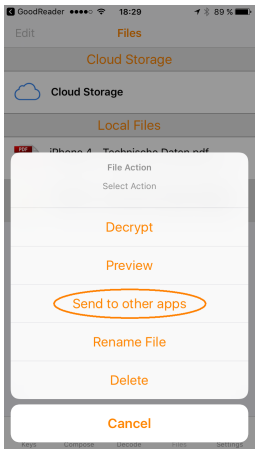
Soll mittels iPGMail eine E-Mail verfasst werden, ist dies über *Compose* direkt möglich. Hier kann zwischen Signatur, Verschlüsselung, Signatur & Verschlüsselung sowie symmetrischer Verschlüsselung mittels Passwort gewählt werden. Über das eingekreiste Plussymbol ⊕ können nur Empfänger ausgewählt werden, für die in iPGMail ein Schlüssel vorhanden ist. Anhänge müssen zuvor in den lokalen Dateibereich *Local Files* kopiert werden, es sei denn, es handelt sich um Fotos oder Dateien aus der Cloud. Diese können direkt übernommen werden.

Ist die E-Mail sendebereit, kann sie über den Export-Dialog mittels *Send Email* abgeschickt werden. Soll auch signiert werden, muss nun das Passwort für den Schlüssel eingegeben oder der Finger für Touch ID aufgelegt werden.



iPGMail-Einstellungen

Die E-Mail wird an die Mail-App weitergereicht, in der das Senden nochmals bestätigt werden muss. Wurde in den *Settings* der Eintrag *Attach PubKey* aktiviert, wird neben dem in PGP/MIME verschlüsselten, eigentlichen E-Mail-Inhalt *encrypted.asc* noch das öffentliche PGP-Zertifikat mitgesendet. In diesem Fall wird also eine „normale“ E-Mail mit zwei Anhängen verschickt. Bei einem kurzen Tipp auf einen Anhang in einer empfangenen E-Mail wird dieser als verschlüsselter Text angezeigt, was natürlich nicht viel nützt. Ein etwas längerer Druck erst öffnet den Auswahldialog von iOS, in dem nunmehr iPGMail ausgewählt werden kann.



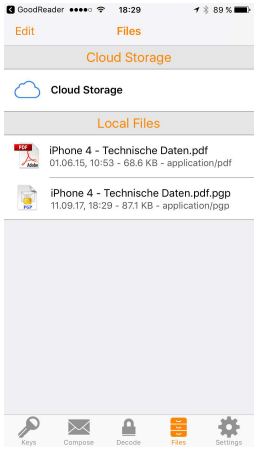
Aus iPGMail an andere Apps senden

Dort angekommen wird die E-Mail entschlüsselt und der nunmehr lesbare Text unter *encrypted.txt* im lokalen Dateibereich abgespeichert. Durch die kryptische Namensgebung, die entschlüsselten Dateien werden einfach durchnummeriert, eignet sich der Bereich nicht wirklich gut als Archiv für empfangene Nachrichten. Die Dateien können aber immerhin umbenannt und natürlich auch an andere Apps weitergegeben werden.

Achtung Falle! Wird über iPGMail eine E-Mail verfasst und an die Mail-App übergeben, wird laut iPGMail-Hilfe eine nicht standardkonforme PGP/MIME-basierte E-Mail erzeugt. Dies führt beispielsweise dazu, dass solch eine Mail im *GpgOL*-Plugin für Outlook nicht entschlüsselt wird. Stattdessen wird nur der Anhang *encrypted.asc* anzeigt. Abhilfe schafft in diesem Fall das Versenden der Mail über einen direkt in iPGMail über *Settings* → *Mail Server Config* definierten SMTP-Server. Um an die entsprechenden Einstellungen zu kommen, muss *Use Default iOS Mail Settings* ausgeschaltet werden.

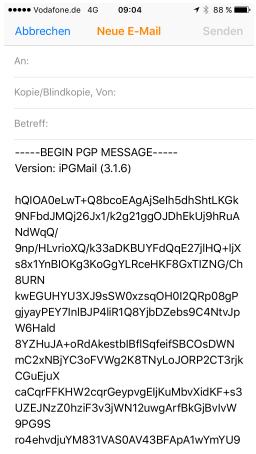
2.5 Datei-Verschlüsselung mit iPGMail

Wird eine Datei über den iOS-Datei-Austausch in den lokalen Bereich von iPGMail kopiert, kann sie auch direkt durch Antippen und anschließendes Auswählen von *Pubkey Encrypt* im *File Action*-Dialog von iPGMail verschlüsselt werden um sie so gesichert beispielsweise auf einem externen Server zu speichern. Eine verschlüsselte Datei bekommt von iPGMail die Dateiendung *.pgp* an den urprünglichen Namen angehängt.



Direktes Verschlüsseln einer Datei

Achtung Falle! Auch wenn in den Einstellungen unter *Default Mode* die Auswahl auf *Sign & Encrypt* steht, wird eine direkt verschlüsselte Datei nicht signiert. Dies ist auch daran zu merken, dass beim Verschlüsselungsvorgang kein Passwort oder eine Touch ID-Anforderung für den Signaturschlüssel abgefragt wird. Es scheint für dieses Problem auch keine Lösung zu geben. Die Einstellung bezieht sich offenbar nur auf E-Mails und nicht auch auf zu verschlüsselnde Dateien.



Mail mit Inline-PGP

Achtung Falle! Immerhin lässt sich eine direkt verschlüsselte Datei problemlos an die Mail-App weiterreichen. Was nun passiert, ist aber wieder nicht im Sinne des Erfinders: Die Datei wird als *Inline-PGP*, also in Textform, in die E-Mail eingefügt, was gerade bei großen Dateien völlig unbrauchbar ist.

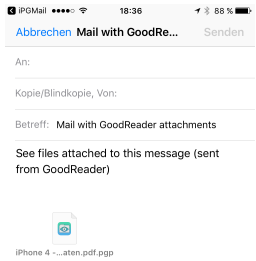
Zumindest gibt es für dieses Problem eine, wenn auch umständliche, Lösung. Der Trick besteht wiederum darin, die zu verschickende Datei zunächst in einer anderen App zwischenzuspeichern, die mit fremden Dateien umgehen kann.

GoodReader

Für iOS empfiehlt sich dafür beispielsweise die App *GoodReader* (derzeit 5,49 €, ab iOS 6) [4], die nicht nur viele verschiedene Dateiearten verwalten und anzeigen kann, sondern unter anderem auch den Zugriff auf Netzwerk-Laufwerke ermöglicht.

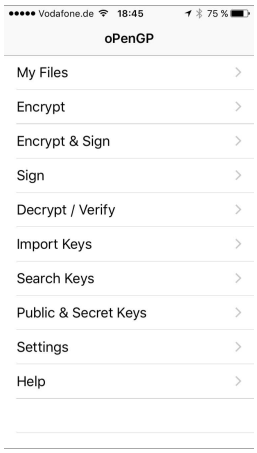
Soll eine solche E-Mail zusätzlich über S/MIME gesichert werden, muss noch die im Abschnitt 1.5 auf Seite 4 beschriebene Prozedur durchlaufen werden. Damit ist es aber auch wirklich getan.

In GoodReader rein...



...und wieder raus

2.6 E-Mails mit oPenGP signieren und verschlüsseln

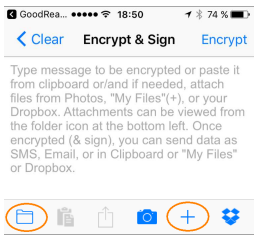


Startbild von oPenGP

Präsentiert sich iPGMail von seiner Optik her schmucklos, ist das Auftreten von oPenGP nur noch spröde zu nennen. Dennoch hat diese App einige Vorteile gegenüber der Konkurrenz, die ihren Einsatz durchaus rechtfertigen können. Besonders der Umgang mit Texten gestaltet sich einfacher und direkter.

Um eine verschlüsselte E-Mail oder einen verschlüsselter Text mittels oPenGP zu verfassen, kann entweder der Bereich *Encrypt* oder der Bereich *Encrypt & Sign* verwendet werden. Für lediglich signierte E-Mails oder Texte gibt es folglich den Bereich *Sign*.

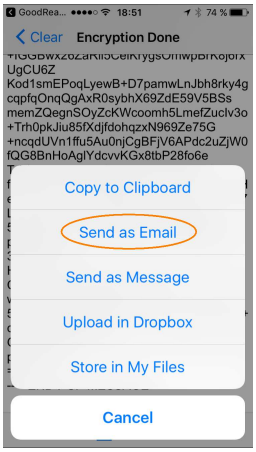
Über das +-Symbol können Anhänge hinzugefügt werden. Über das Ordner-Symbol können diese Anhänge überprüft und auch wieder entfernt werden.



Verschlüsseln & Signieren mit Anhang

Nach Eintippen des Textes und eventuellem Hin-

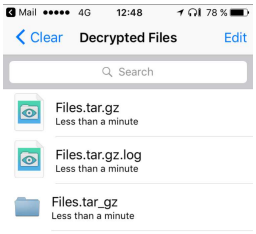
zufügen von Anhängen wird durch Antippen von *Encrypt* und *Sign* der Verschlüsselungsvorgang gestartet und bei Bedarf das zum Schlüssel gehörende Passwort abgefragt. Auch hier ist die Alternative über Touch ID möglich. Anschließend gilt es auszuwählen, was mit dem Ergebnis passieren soll. *Send as Email* dürfte dabei die häufigste Entscheidung sein.



Aus oPenGP an andere Apps senden

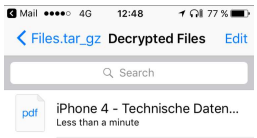
Verschlüsselte E-Mail mit Anhang

Die E-Mail wird an die Mail-App weitergereicht, in der das Senden wie immer in solchen Fällen nochmals bestätigt werden muss. Auch hier sieht es wie schon im Abschnitt 2.5 auf Seite 7 beschrieben zunächst so aus, als ob eine mit *Inline-PGP* formatierte Mail erzeugt wurde. „Inline“ ist aber nur der E-Mail-Text verschlüsselt worden, eventuelle Anhänge werden als *Files.tar.gz.pgp* mit angehängt. Bei Empfang einer solchen Mail muss der verschlüsselte Text zunächst kopiert und oPenGP gestartet werden. In der App wird der kopierte Text in den Bereich *Decrypt / Verify* automatisch übernommen und kann anschließend dort entschlüsselt werden.



Entschlüsselter Anhang

Achtung Falle! Mittels *.tar.gz* erstellte Anhänge können nicht von jedem Programm entpackt werden. Während es in der Desktop-Welt wohl wenig Schwierigkeiten damit geben wird, sieht die Lage in der Welt der Mobilgeräte anders aus. iPGMail beispielsweise ist nicht in der Lage, diese Anhänge vernünftig zu verarbeiten. Auch GoodReader kann damit nichts anfangen.



Datei im tar.gz-Archiv

Naturgemäß keine Probleme hat oPenGP selbst. Es entschlüsselt den Anhang nach Passwort-Eingabe und stellt den eigentlichen Inhalt im Ordner *Files.tar_gz* zur Verfügung. Der Inhalt kann bei bekanntem Dateiformat direkt in oPenGP betrachtet oder an andere Apps gesendet werden. Wird der Anhang nicht in einer anderen App gesichert, geht er beim Schließen von oPenGP verloren, was sich durchaus als Sicherheitsmerkmal verstehen lässt.

2.7 Datei-Verschlüsselung mit oPenGP

Wird eine Datei über den iOS-Datei-Austausch in den lokalen Bereich von oPenGP kopiert, kann sie auch direkt durch Antippen und anschließendes Auswählen von *Encrypt & Sign* im Bereich *My Files* von oPenGP verschlüsselt werden um sie so gesichert beispielsweise auf einem externen Server zu speichern. Eine verschlüsselte Datei bekommt von oPenGP die Dateiendung *.pgp* an den urprünglichen Namen angehängt.

Achtung Falle! Eine direkt im *My Files*-Bereich verschlüsselte Datei kann nur wieder über *Decrypt / Verify* entschlüsselt, umbenannt oder zum Anbieter *Dropbox* hochgeladen werden. Da die App seit Dezember 2015 nicht mehr aktualisiert wurde, wird dieser Weg wohl bald versperrt sein. Ein lokales Versenden an eine andere App ist nicht möglich, auch nicht an die Mail-App. Es bleibt also nur die Möglichkeit, eine Datei wie bei einer E-Mail zu verschlüsseln und anschließend an sich selbst zu versenden.

A Zertifikatsbeantragung für S/MIME

Die Beantragung eines persöhnlichen Zertifikats für den Verschlüsselungsstandard S/MIME wird im Folgenden am Beispiel von Comodo [5] beschrieben. Bei anderen Zertifizierungsstellen sollte der Vorgang im Wesentlichen ähnlich ablaufen.

Über den Pfad *https://www.comodo.com* → *Personal* → *Free Personal Email Certificate* → *Free Email Certificate: Free Download* kann das Online-Formular für den Zertifikatsantrag von Comodo erreicht werden. Hier müssen lediglich die entsprechenden Daten ausgefüllt und den Geschäftsbedingungen zugestimmt werden.

S/MIME-Zertifikat von Comodo beziehen

Im folgende Schritt werden vom Kryptografie-Modul des verwendeten Internet-Browsers ein öffentlicher Schlüssel und der dazu gehörende private Schlüssel erzeugt. Da der private Schlüssel den eigenen Rechner nicht verlässt, sondern im Kryptografie-Modul des Browsers verbleibt, muss der gesamte Vorgang mit dem selben Browser auf dem selben Rechner vollständig durchgeführt werden.
Nach den gemachten Angaben generiert das Webportal also sowohl das Schlüsselpaar, bestehend aus öffentlichem und privatem Schlüssel, als auch den zugehörigen elektronischen Zertifikatsantrag, bestehend aus Antrag und öffentlichem Schlüssel. Der Zertifikatsantrag wird durch den Browser an Comodo.com versendet.
Die Überprüfung der E-Mail-Adresse erfolgt wie bei anderen Portalen auch durch Zusenden eines entsprechenden Links, welcher im E-Mail-Programm innerhalb einer bestimmten Zeit angeklickt werden muss. Über die Verlinkung kann anschließend das Zertifikat heruntergeladen werden. Wichtig ist wie oben beschrieben, dies mit dem selben Internet-Browser zu tun, mit dem auch das Schlüsselpaar erzeugt wurde. Über den in Abschnitt 1.2 auf Seite 1 beschriebenen Vorgang kann das Zertifikat gesichert und auf das iOS-Gerät übertragen werden.

B Aktualisierung nach Fertigstellung des Berichts

Oktober 2017:

Mit iOS 11 und iOS 11.0.1 hat sich ein Fehler in der Behandlung von Anhängen bei S/MIME-verschlüsselten E-Mails eingeschlichen, der dazu führt, dass verschlüsselte Anhänge nicht vollständig heruntergeladen werden und damit unbrauchbar sind. Dieser Fehler wird mit der Aktualisierung auf iOS 11.0.2 korrigiert. Versionen vor iOS 11 sind nicht betroffen.

Fehlerkorrektur für S/MIME-Anhänge mit iOS 11.0.2

Referenzen

[1] *Enigmail – A simple interface for OpenPGP email security.* www.enigmail.net, 2017.

[2] *Gpg4win – eine sichere Lösung zum Verschlüsseln und Signieren von E-Mails, Dateien und Ordnern unter Windows.* www.gpg4win.de, 2017.

[3] *iPGMail.* ipgmail.com, 2017.

[4] *GoodReader.* www.goodreader.com, 2017.

[5] *Comodo – Creating Trust Online.* www.comodo.com, 2017.