Email and File Encryption on iOS with S/MIME and PGP

Peter Tondl

Email and file encryption are no longer an issue in the desktop world. Either you do it or not. In any case, technology rarely fails. All relevant mail clients support native encryption with S/MIME. For encryption with PGP freely available software exist like *Enigmail* [1] or *Gpg4win* [2]. The latter not only provides a plugin for Outlook, but also allows convenient encryption of files via the context menu of a file explorer. However, things are different in the mobile world. For S/MIME as well as PGP a few hurdles have to be overcome and traps have to be avoided. How this can be done is explained by the example of iOS in the following report.

1 Email and S/MIME

First the good news: Apple's email client for iOS already supports the encryption standard S/MIME by itself, so no further apps are needed.

1.1 Requirements

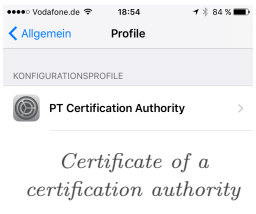
In order to create an encrypted email, a sender first requires an own X.509 certificate. X.509 is the name for the standard, after such certificates are established. A certification body thus confirms in cryptographically secured form that, for example, an email address of a certificate applicant belongs to a submitted public key. Together with the private key, which no one else knows and is allowed to know except the owner, all requirements are met on the part of the sender to be able to exchange encrypted emails. How to create the required secure encryption key pair, consisting of a public key and a secret private key, and how to submit the public key to a certification authority to get a full X.509 certificate along with the signature of that certification authority is described in the appendix.

1.2 Get your own key pair securely on the iOS device

Certificate export from Firefox

Once the process for creating the personal X.509 certificate has been successfully completed, the certificate together with the generated key pair is typically located in the security module of the used Internet browser. From this it must first be exported for its use with iOS. How this exactly works depends on the browser, but the principle is always the same. If Mozilla's Firefox is used, the path to your own certificate will be displayed, for example, via *Tools* → *Options* → *Advanced* → *Certificates* → *View Certificates* → *Your Certificates*. Once there, the certificate can be exported via *Backup...* to a PKCS12 file (*.p12).

Certificate import into the Profiles section



Certificate of a certification authority

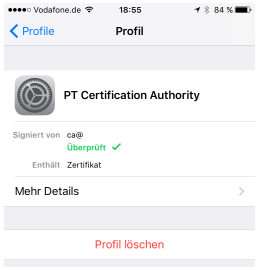
Trust setting for certification authorities

It is highly recommended to use a strong password for the export of the certificate because the private key has to be protected against unauthorized access. This password is no longer needed after importing the certificate into iOS, so it should be longer and more complex than usual. Although according to pure teaching a transfer of the *own* certificate – and thus of the private key as well – via E-Mail is prohibited, this is the usual way. The reason for this is certainly the cumbersome handling of files under iOS, in which email is often the only way to exchange data between an iOS device and its environment. In order to at least minimize the risk on the transmission path, it is recommended to send the exported certificate to yourself so that the email (hopefully) does not leave the provider's server. If the transmission path to and from the email server is actually secured via SSL, as it is usually the case today, an interception of the email by a third party can also be prevented in this way. After importing the certificate, the email should be deleted, in the folder for received mails *as well as* in the folder for sent mails.

1.3 Email account settings for use with S/MIME

The import of your own certificate is started by tapping the email attachment. iOS prompts the user to enter the password specified above during export. After successful typing, the certificate including the private key will be imported into iOS and will be visible in the Profiles section under *Settings* → *General* → *Profiles*. In this area, the so-called root certificates of certification bodies are stored, which can also be imported via the path described above. Such an additional import is always necessary if iOS itself does not know the certification authority, for example, in case of a self-created certificate signed by an own certification authority. This happens usually with larger companies that operate their own certification infrastructure.

Certification authorities with full user confidence



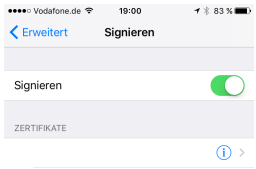
Caution! Since iOS 10.3, it is no longer sufficient just to import a certificate from a certification authority as in previous versions. Certificates certified by such a certification authority will not be considered valid until the root certificate has been given full trust in the *Settings* → *General* → *About* → *Certificate Trust Settings* settings.

When a user has enabled the trust setting of a certificate, certificates derived from this root certificate are automatically considered valid since they have been certified by the root certificate. Separate trust settings for subordinate certificates are therefore not required.

Manually added certificates can be removed from the certificate store at any time using the *Remove Profile* command. If a root certificate is removed, all certificates that depend on it also lose their validity and can no longer be used to encrypt new emails. However, decryp-

S/MIME enabling

ting of already existing emails is still possible, even if a warning about the non-existent validity is displayed.



Sign by default

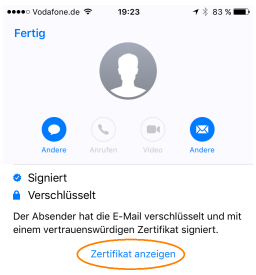
Encrypt by default

Enabling S/MIME for the email account however, is not quite enough. If emails are also to be signed and possibly encrypted when sending, the signature or encryption for outgoing emails must also be entered under *Settings* → *Mail* (up to iOS 10) *Accounts & Passwords* (from iOS 11) → *Accounts* → "Name of the email account" → *Account* → *Advanced* to be activated.

Unfortunately, this can not be set individually for each email. Although the encryption for a particular email can be turned off if it has been previously turned on for the email account in general, the reverse case is not possible. Also, signing can not be turned off when it is turned on for the account. In this case, signing always takes place.

Sign by default as well as *Encrypt by default* is only possible if your own, valid certificate is present. All existing own and valid certificates are offered for selection in the respective settings pages. If an assignment is clearly possible, iOS chooses a certificate by itself, which is then marked with a tick.

Receiving of an encrypted and signed email



Displaying of a received certificate

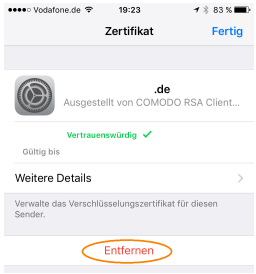
Installing of a received certificate

1.4 Sign and encrypt emails with S/MIME

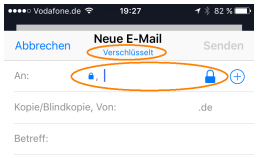
Once all the certificates have been put on board and all the switches have been set, it can finally start. At least with the signing of own mails. To enable encryption an appropriate certificate of the communication partner is still missing. Without this it is not possible, because the mail client does not have a key for which it should encrypt a message. The easiest way to get such a key is to get a signed email from the desired mail partner. An S/MIME signed mail contains always the sender's certificate, which can be taken over in the own certificate store by a few taps. From now on, encryption to this email partner is possible as long as its certificate is valid.

Caution! Certificates of communication partners are not stored in the profile area described above. Moreover, there is no other area within iOS, either in the Settings or elsewhere, where installed certificates can be accessed. This leads to an unpleasant effect: If the certificate to an email address is to be replaced by a new certificate, it can apparently be installed as described above, but in fact the old certificate will not be overwritten by the new one. Thus, there is no possibility to exchange the certificate of an email partner or just even to remove it!

IMAP folder for received certificates



Removing of a received certificate



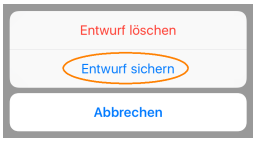
Composing of an encrypted email

Fortunately, there is a simple trick to avoid this misery. When the properties of a certificate attached to an email that already exists in the iOS storage are opened, iOS will no longer offer the option to install the certificate, but will now be able to remove it. Thus, for each installed certificate of a communication partner only one signed email is required in order to remove this certificate again from iOS own memory. To

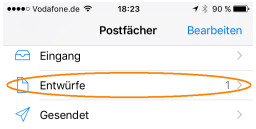
always be able to access a suitable email for this purpose, a separate folder can be created in the email account in which all emails, originally used for installation, are stored for later use.

If this hurdle is taken, it can finally start. Depending on the chosen recipient, the mail app displays a lock symbol to indicate whether it is possible to encrypt (signing is always possible and will always be done if *Sign by default* is enabled). At the same time, with possible encryption, that is, there is a valid certificate for each (!) recipient, a note *Encrypted* appears in the header of the email. In order to switch off the encryption for this email, all you need to do is to click on the large closed lock symbol, which then opens. If encryption is not possible, this is made very clear in the header area by recipients appearing in red and the note *Unable to Encrypt*. The most common reason for this is probably that the certificate of an email partner either does not exist or is no longer valid.

1.5 Email encryption from other apps



Securing of an email draft



Recalling of an email draft

Enabling of local saving for drafts and trash

Actually, this section should not exist. But for some unknown reason, email encryption using S/MIME can not be enabled in the mail app when it was called from another app, such as by the *Send To* command (as far as iOS 11.1.2).

Caution! In this case, the email is always sent unencrypted and unsigned. Fortunately, there exists at least an alternative approach for this problem as well, until it may be remedied by Apple in the distant future. The trick is not to send the email after it has been composed, but to cancel the process. The mail app then asks if the draft should be saved or deleted. Of course, *Save Draft* is the right choice here, because the mail is supposed to reach its recipient after all. Then, the email draft has to be recalled from the drafts folder. The difference to direct sending is that encryption of the email is now miraculously possible.

There remains a problem to be solved that an unencrypted draft in case of saving, at least by using IMAP as email protocol, first lands on the server of the provider of this service. Since this is generally not intended, the locations for the draft and trash folders have to be moved from the email server to the iOS device under *Settings* → *Mail* (to iOS 10) *Accounts & Passwords* (from iOS 11) → *Accounts* → "Name of the email account" → *Account* → *Advanced*. Thus, no unencrypted draft and no more unencrypted and delete email leaves the device.

2 Email and file encryption with PGP

While S/MIME is already tightly integrated with Apple's mail app, using PGP as an encryption system is a bit more of a challenge. However, this is rewarded with the added functionality of being able to cryptographically secure files via email regardless of their delivery, since PGP, unlike S/MIME, is not restricted to use by email.

2.1 Requirements

For iOS, there are two commercial apps (currently 2.29 € and 5.49 €), both of which, as is often the case, have their pros and cons.

iPGMail

oPenGP

As system requirement, *iPGMail* needs iOS 8.0 or later [3]. For older versions of iOS, the last compatible version of the app can also be downloaded when the current version of the app has been previously installed on an iOS 8 enabled device using the same Apple ID. However, this older version crashes on iOS 7 without a last goodbye. The latest update of the app was in July 2017.

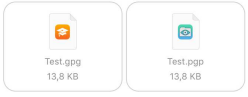
oPenGP is much more modest, iOS 7 or newer is absolutely sufficient. Of course, this may also be due to the fact that the app has not been updated since December 2015. Nevertheless, it is not a pure 32 bit app, so it runs on iOS 11 without any problems.

2.2 Get your own key pair securely on the iOS device

Both apps support PGP private key transfer via iTunes data exchange and USB cable. This enables a particularly secure transfer of the private key from a PC to an iOS device. If the above described encryption of emails using S/MIME is possible, this method can also be used safely. In general, however, it must always be kept in mind that the use of a particularly "valuable", since possibly used for years and thus known and accepted by many communication partners, PGP key on a mobile device is certainly not without problems. After all, the validity of such a key is not limited to mostly one year, as it is the case with S/MIME. Proper protection of the iOS device against unauthorized use is therefore mandatory.

2.3 Email and PGP

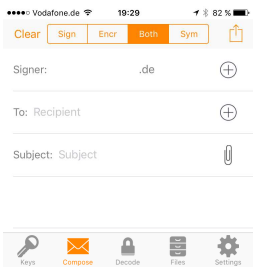
Since Apple does not allow extensions in its mail app, but also does not allow an exchange of the default email app, iPGMail as well as oPenGP can only be used as a "side-app": If a PGP email was received, which means either a PGP/MIME encoded email or a "normal" email with a PGP encrypted file attachment, this attachment can only be accessed properly by tapping in the email and thus send it to iPGMail or oPenGP.



PGP attachments

Caution! PGP attachments are not necessarily recognizable as such in an email. For example, depending on your own device configuration, it may well happen that an attachment with the extension *.gpg* as an *iTunes U* file and an attachment with the extension *.pgp* as a *GoodReader* file can be displayed. In case of doubt only trying out or looking closely helps.

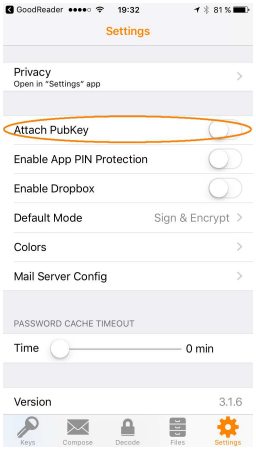
2.4 Sign and encrypt emails with iPGMail



Write an email

If an email is to be written using iPGMail, this is directly possible via *Compose*. Here you can choose between signature, encryption, signature & encryption as well as symmetric encryption via password. The circled plus symbol ⊕ can only be used to select recipients who have a key in iPGMail already. Attachments must first be copied to the local *Files* section, unless they are photos or files from the cloud. These can be taken over directly.

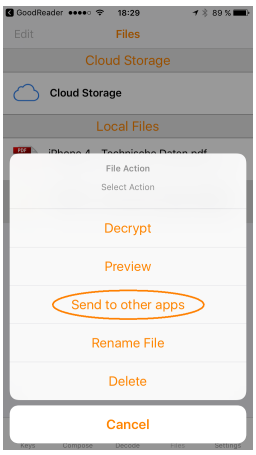
If the email is ready to be sent, it can be sent via the export dialog *Send Email*. If you also want to sign, you must now enter the password for the key or place the finger for Touch ID.



iPGMail settings

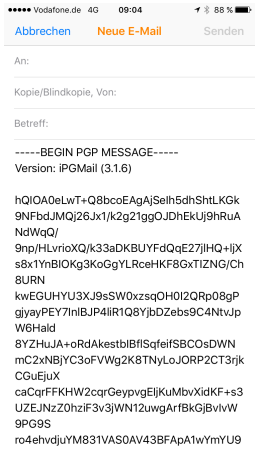
The email is forwarded to the email app, in which the sending must be confirmed again. If the entry *Attach PubKey* has been enabled in the settings, the public PGP certificate will be sent along with the encrypted PGP/-MIME email content *encrypted.asc*. In this case a "normal" email with two attachments will be sent. A quick tip on an attachment in a received email displays it as encrypted text, which of course does not do much good. A little longer pressure opens the selection dialog of iOS, in which now iPGMail can be selected.

Once there, the email is decrypted and the now readable text is stored under *encrypted.txt* in the local file area. Due to the cryptic naming, decrypted files are simply numbered, the area is not really good as an archive for received messages. After all, files can be renamed and of course also be passed on to other apps.



Send from iPGMail to other apps

Caution! If an email is written via iPGMail and passed to the email app, iPGMail generates a non-standard PGP/MIME based email, according to its own help text. This leads, for example, to the fact that such a mail is not decrypted in the *GpgOL* plugin for Outlook. Instead, only the attachment *encrypted.asc* is displayed. A solution in this case is to send the mail via an SMTP server, defined directly in iPGMail via *Settings* → *Mail Server Config*. In order to get to the appropriate settings, the entry *Use Default iOS Mail Settings* must be turned off first.



Email with inline PGP

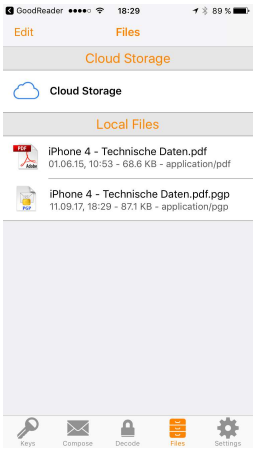
Caution! After all, a directly encrypted file can easily be forwarded to the mail app. What happens now, but again not in the sense of the inventor, is: The file is inserted as *Inline PGP*, thus in text form, in the email, which is completely unusable, especially for large files.

At least there is a solution to this problem, although cumbersome. The trick, once again, is to first cache the file to be sent in another app that can handle foreign files.

GoodReader

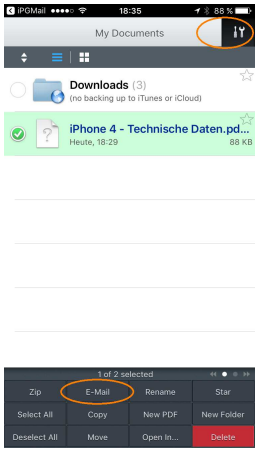
For iOS, for example, the app *GoodReader* [4] (currently 5.49 €, from iOS 6 onwards) is recommended, which not only manages and displays many different types of files, but also provides access to network drives, among other things.

2.5 File encryption with iPGMail

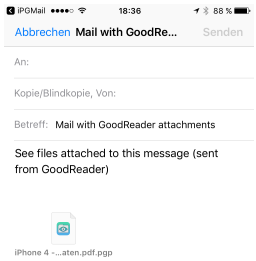


Direct file encryption

If a file is copied to iPGMail's local area via iOS file exchange, it can also be encrypted directly by tapping and then selecting *Pubkey Encrypt* in the *File Action* dialog of iPGMail, for example, to save it to an external server. An encrypted file gets the file extension *.pgp* appended to the original name by iPGMail.



Into GoodReader...

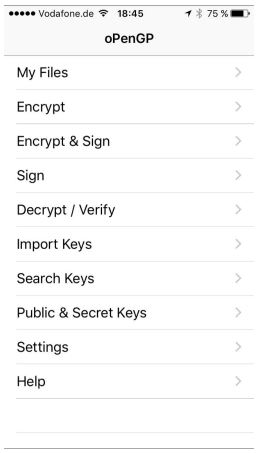


...and out again

Caution! Even if the settings under *Default Mode* are set to *Sign & Encrypt*, a directly encrypted file will not be signed. This is also to be seen that during the encryption process, no password or a Touch ID request for the signature key is queried. There also seems to be no solution to this problem. The setting apparently only applies to emails and not to files to be encrypted directly.

If such an email is also to be secured via S/MIME, the procedure described in section 1.5 on page 4 must be followed. But that's really it.

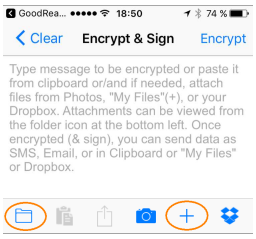
2.6 Sign and encrypt emails with oPenGP



Start picture of oPenGP

If iPGMail is unadorned by its appearance, the appearance of oPenGP is only brittle. Nevertheless, this app has some advantages over the competition, which can well justify its use. Especially handling of texts is easier and more direct.

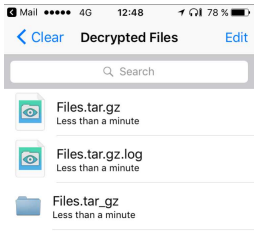
To write an encrypted email or encrypted text using oPenGP, either the *Encrypt* area or the *Encrypt & Sign* area can be used. For only signed emails or texts there is consequently the area *Sign*.



Encrypt & Sign with Attachment

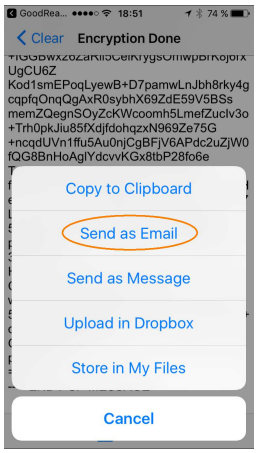
Attachments can be added via the + symbol. The folder icon can be used to check and remove these attachments.

After typing in text and adding attachments, tap commands *Encrypt* and *Sign* to start the encryption process and, if necessary, query the password associated with the key. Again, the alternative via Touch ID is possible. Then you have to choose what should happen to the result. *Send as email* will be the most common decision.



Decrypted attachment

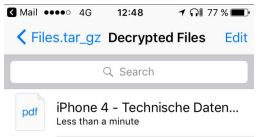
The email is passed to the email app, in which the sending must be confirmed again, as always in such cases. Once more, as described in section 2.5 on page 6, it looks as if a mail formatted with *Inline PGP* has been created. Nevertheless, only the email text has been encrypted "Inline", any attachments are appended as *Files.tar.gz.pgp*. When receiving such a mail, the encrypted text must first be copied and oPenGP started. In the app, the copied text is automatically transferred to the *Decrypt / Verify* area and will then be decrypted there.



Send from oPenGP to other apps

Encrypted email with attachment

Caution! Attachments created with *.tar.gz* can not be unzipped by a lot of programs. While there will not be much trouble in the desktop world, the situation in the mobile world is different. For example, iPGMail is unable to reasonably handle these attachments. Even GoodReader can not do anything with it.



File within tar.gz archive

Of course, no problem has oPenGP itself. It decrypts the attachment after password entry and provides the actual content in the folder *Files.tar_gz*. The content can be viewed directly in oPenGP or sent to other apps if the can handle the current file format. If the attachment is not saved in another app, it will be lost when closing oPenGP, which can be considered as a security feature.

2.7 File encryption with oPenGP

If a file is copied to the local area of oPenGP via iOS *Send to* dialog, it can also be encrypted directly by tapping and then selecting *Encrypt & Sign* in the *My Files* area of oPenGP to save it, for example, to an external server. An encrypted file gets the file extension *.pgp* appended to the original name by oPenGP.

Caution! A directly encrypted file in the *My Files* area of oPenGP can only be decrypted

again via *Decrypt/Verify*, renamed or uploaded to the provider *Dropbox*. Since the app has not been updated since December 2015, this last way will probably be blocked soon. A local sending to another app is not possible, not even to the mail app. So there is only the possibility to encrypt a file like an email and then send it via email to yourself.

A Certificate application for S/MIME

The application for a personal certificate for the encryption standard S/MIME is described below using the example of Comodo [5]. For other certification authorities, the process should be essentially similar.
Via the path *https://www.comodo.com* → *Personal* → *Free Personal Email Certificate* → *Free Email Certificate: Free Download* an online form for the certificate request from Comodo can be reached. All you have to do is to fill out the relevant data and agree to the terms and conditions.

Get an S/MIME certificate from Comodo

In the following step, a public key and the associated private key are generated by the cryptography module of the Internet browser used. Since the private key does not leave your own computer, but remains in the browser's cryptographic module instead, the entire process must be completed on the same computer with the same browser.
According to the information provided, the web portal thus generates both the key pair, consisting of public and private key, and the associated electronic certificate application, consisting of application and public key. The certificate request will be sent by the browser to Comodo.com.
As with other portals, the verification of the email address is also carried out by sending a corresponding link, which must be clicked in the email client within a certain time. The certificate can then be downloaded via the link. It is important, as described above, to do this with the same internet browser with which the key pair was also created. The process described in section 1.2 on page 1 can be used to save the certificate and transfer it to the iOS device.

B Update after completion of this report

October 2017:
With iOS 11 and iOS 11.0.1, there has been a bug in handling of attachments to S/MIME encrypted emails, which means that encrypted attachments are not fully downloaded and therefore unusable. This bug is corrected with the upgrade to iOS 11.0.2. Versions prior to iOS 11 are not affected.

Bug fix for S/MIME attachments with iOS 11.0.2

References

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

[2] *Gpg4win – a secure solution for file and email encryption.* www.gpg4win.de, 2017.

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

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

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