Autocrypt automates email cryptography

# presented

**Jan Bundesman**

Autocrypt wants to make it easier for users to manage cryptographic keys. Several clients are already implementing the standard.

**-TRACT**

- As end-to-end email encryption remains the exception, several initiatives are working to simplify it.
- Version 1.0 of the Autocrypt specification was released in December 2017. It should make key management easier and automatically encrypt it if the communication participants want it.
- First clients have integrated Autocrypt.

**rating**

⊕ Distribution of cryptographic keys without a central architecture

⊕ Synchronization of multiple clients via setup messages

⊖ no authentication of assignment of keys to users

A distribute keys aggressively, but only encrypt gently - one of the principles of the Autocrypt specification. The document, version 1.0 of which was published at the end of 2017, defines rules for encrypting message content and for key exchange. It was specially developed with usability in mind. The Autocrypt developers, among others, believe that this is not the case with the current methods for e-mail encryption. They refer to the two common methods OpenPGP and S/MIME.

So far, OpenPGP key management has been entirely in the hands of the users. They have to download the files from a public key server for each contact. Since these servers do not offer any mechanisms to check the stored identity, this is done out of band, in a separate channel. It is conceivable, for example, that someone – intentionally or by mistake – stores a public key under an incorrect e-mail address. Therefore, users have to put additional work into trust management. With GnuPG, keys are given a trust level. You should choose the highest level if you are convinced of the authenticity of the key. This applies, among other things, if the identity has been established by comparing the fingerprint of the key in the presence of the owner. This can also be uploaded to the server. There it is stored who vouches for the authenticity of the entry. If there is a trustworthy body, you can use a key without a personal check - that's the idea behind the Web of Trust.

S/MIME solved this with a hierarchical structure: In principle, users buy trust from a designated authority that signs the key and issues a digital certificate. This authority has in turn certified a higher authority, which creates a chain of certificates at the top of which a trustworthy partner should be at the latest. A number of providers, including free ones, provide certificates.

A client that has implemented Autocrypt uses a public key method like PGP and S/MIME – currently RSA with 3072-bit keys. There is no central instance involved, the software handles the key management in the background without the user having to intervene - that is the core idea. The developers see the fact that users receive keys directly from the owner as the basis for "Trust on first use" (TOFU). Ideally, the usual suspects, Alice and Bob, exchange two emails and are then in possession of the public key of their respective communication partner. All future messages are encrypted and signed. Autocrypt automatically trusts the digital signature.

Unlike OpenPGP key servers, users only see other public keys if they came in via an email that their provider checked. Many email providers intercept phishing emails using special methods such as DKIM. So also those emails that try to foist a false autocrypt key with a fake sender.

The developers of Autocrypt are thus pursuing similar goals to those of Pretty Easy Privacy - simpler trust management. The latter, however, provide a library that can be integrated into various mail clients. Autocrypt, on the other hand, is based on a specification that mail client developers implement themselves. The project describes itself on the website as a "social and technical effort": Therefore, the participants try to get developers on board in particular, so that implementation issues can be incorporated at an early stage.

## In the head

Instead of a central key management, Autocrypt uses an active one-Key distribution in email header. *There, in a separate Autocrypt entry*, you will find the e-mail address as identification for the key, which is also placed there. You can also convey your preference under which circumstances cryptographic operations should be used:

```
Autocrypt: addr=a@b.example.org; [
[prefer-encrypt=mutual;] keydata=AA5E64
```

*keydata* contains a key in the format of an OpenPGP "Transferable Public Key". An Autocrypt client reads this information from incoming emails. For each communication partner, it saves whether encryption is possible and desired. When composing an email, it decides autonomously whether plain-text or encrypted content is to be sent. According to the principle of only encrypting carefully, Autocrypt only recommends this in three cases:

1. The sender explicitly requests this.
2. You reply to an encrypted mail.
3. All communication participants use Autocrypt and have stored in their settings - i.e. announced via mail header - that they prefer encrypted mails.

## Full access to the mail header

Since all Autocrypt logic happens in the header, the mail client needs to be able to browse and customize it. For web interfaces, the providers have to adapt their interface. However, the developers of the Mailvelope browser extension, for example, are also working on integrating Autocrypt in order to implement end-to-end encryption for webmail clients.

### What software implements Autocrypt?

Since Autocrypt is just a specification, mail client developers have to implement it themselves, while Pretty Easy Privacy (pEp), for example, provides plugins for a variety of clients. As a result, the number of Autocrypt clients is still relatively manageable. This includes the Thunderbird plugin Enigmail. There are also some tools for PCs that can be embedded in other software, such as *muscrypt* , *pyac* , and *gmime* .

Android users can also use Autocrypt. The mail client K-9 Mail currently allows this. In this case, OpenKeychain takes over the key management. *delta.chat* is a messenger that can be obtained from the alternative app store F-Droid. It uses the Telegram UI but not its backend. Instead, *delta.chat* is based on an e-mail infrastructure: incoming messages are received via IMAP push and dispatched via SMTP.

In fact, the number of mail clients with Autocrypt integration still has to grow for the project (see the box "Which software implements Autocrypt?").

In order to increase usability, the developers decided to use the header as the transport medium instead of an attachment. Among other things, this was done to avoid users being deterred without the necessary knowledge – for example by an attachment that they mistake for spam or malware. After all, the communicated content is more important than encryption, which is why the threshold for not using cryptography is relatively low.

So that previously secret content is not suddenly revealed, Autocrypt clients should not include any passages when replying to encrypted emails - for example as a quote - if the reply is written in plain text.

Encryption with 3072-bit RSA keys is currently planned. The client should create two pairs of these: one for signing and self-certifying, one for encryption. An encrypted mail in the sense of Autocrypt is always both encrypted and signed.

**If you want to operate several clients with Autocrypt, the settings can be transferred via setup message. This is an AES-encrypted email. Users have to enter the key manually on the second device.**
Photo: Bjorn Petersen

The specification provides setup messages for operation with multiple clients or on multiple devices. These are emails with all the information to set up Autocrypt, which are AES-128 encrypted. The key required for this consists of 36 digits. To whomyou have to transfer it manually to activate a new application. The clients have their own input window for this.

The key can also be imported into any PGP installation. To do this, the software must be given the content of the setup mail for decryption. The secret key created by Autocrypt is packed in it. Users need the 36 digits mentioned to decrypt – including the hyphens. In this way, the Autocrypt-encrypted messages can also be read on other clients that only support OpenPGP. Of course, these do not make any adjustments to the mail header, so there is no guarantee that the encryption preferences will be retained.

However, Autocrypt is not undisputed. The main criticism is the underlying opportunistic security from RFC 7435. This provides "some protection most of the time". Plain text communication is the basis, the participants only activate encryption after at least two e-mails – as is the case with pEp, by the way. The developers of Autocrypt see this as a strength of their approach, because in their eyes the number of encrypted emails is increasing and this ultimately leads to more widespread end-to-end encryption. However, the approach primarily protects against passive attacks, i.e. large-scale recording of mail traffic.

## Exchange the key on the mail server

### Who else wants to simplify key distribution?

After decades of pure teaching and lamenting about complicated central key infrastructures, there are now four projects dedicated to this topic. The three projects alongside Autocrypt also originally come from German-speaking countries and are mainly developed there.

GnuPG has developed the concept of Web Key Directories over the last two years. This still requires a server infrastructure. Mail clients should automatically fetch public keys from the servers when they compose a message - if the GnuPG developers have their way, the mail providers will take over the operation of the necessary infrastructure and also implement it in their web clients.

Pretty Easy Privacy (pEp) implements a peer-to-peer key exchange network. Encoded messages are sent encrypted according to the OpenPGP standard if no other method is available. If there is a direct online channel between the communication partners, the message is sent via the P2P network or via an anonymization platform such as Qabel. Unlike Autocrypt, pEp always tries to encrypt emails. Plain text messages are only intended as an emergency solution.

A project funded by the Federal Ministry of Education and Research is called "Trustworthy Distribution of Encryption Keys" (Project VVV). University institutes and a mail provider, mailbox.org, are involved. The management lies with the Fraunhofer Institute for Secure Information Technology. According to the VVV concept, public keys are located at a defined location that clients can query from the DNS entry of the email domain in the form of a "Service Resource Record". This is signed using DNS Security Extensions (DNSSEC). With a complete chain of signature keys up to the root domain of the DNS, users should be able to trust keys obtained in this way.

All four approaches should run in the background. Mail clients obtain or collect keys automatically. Manual procurement is no longer necessary, as is manually setting the trust level - because it is either given or the specification does not.

Specification 1.0 does not provide for authentication of the keys, giving active attackers a leverage point. If they have an influence on the operator of the mail server, they can exchange the headers of the e-mails sent. Due to the lack of cryptographic validation of the keys and TOFU, this is not visible to the recipient. The authors of the privacy manual therefore even recommend completely disabling Autocrypt in Enigmail 2.0. The operators of the mail provider mailbox.org warn of the "deceptive security" offered by Autocrypt. You yourself rely on the Trusted Distribution of Encryption Keys (VVV) project, which mailbox.org is also working on.

The Autocrypt community is planning a solution for one of the upcoming releases. In their eyes, man-in-the-middle attacks are already making it more difficult because DKIM signatures on the Autocrypt header require attacks in many places. Additionally, they recommend out-of-band verification for anyone wanting to protect against active attacks. They point out that the Messenger Signal without key verification is just as vulnerable to an active attack as Autocrypt.

## Conclusion

Autocrypt has reached a usable stage but, like Pretty Easy Privacy, currently suffers from a lack of ready-made clients. In order to become interesting for users, however, software and sufficient distribution are necessary prerequisites.

In view of the alternatives that are emerging at the same time, the question of compatibility with the other approaches also arises. In addition, existing key collections should not simply lose their value.

The criticism of Autocrypt does not relate to its implementation, but to fundamental design decisions such as not having a cryptographic, automated verification of the keys. This can only be used to generate cryptographic noise that protects against passive threats. However, from the developer's point of view, scenarios for active attacks on Autocrypt are not easy. And if you want to protect yourself from active attackers, you should continue to rely on explicit verification of the data. ( jab@ix.de )

All links: ix.de/ix1805064

**Read comments (4 posts)**

write a letter to the editor | Download article as PDF | Share on Facebook | Share on Twitter