

# EndCryptor

Version 2.5.4 [www.endcryptor.com](http://www.endcryptor.com). Product of Enternet Inc., Finland.

**TIP: Use Bookmarks for navigation**

## Contents

Overall description	2
Quick start guide	3
Useful tips	4
Security features explained	6
Tutorial on public keys	11
EndCryptor, S/MIME and PGP under attack	14
The risks of SSL	15
Cryptographic technical details	21
To: Really security conscious user	28
Avoid security through obscurity	29
Obtaining a license	30

Date of this document: February 12, 2019

## **Overall description**

### **Superior protection for the real world**

EndCryptor protects old encrypted emails even if a hacker gets current encryption keys. Recently viruses (which were undetected for a decade) were found that stole encryption keys of known email encryption solutions – thus enabling the decryption of earlier messages. EndCryptor is designed to protect old messages and also to recover from attack – the attacker will lose the ability to decrypt new incoming messages.

### **Easy to use**

No knowledge of cryptography is required. The user interface is similar to a typical email client. User's current email account is used to deliver the encrypted emails.

### **End to End Encryption**

The email is encrypted at sender's computer and decrypted at receiver's computer. Only the true receiver can decrypt the email.

### **Quantum attack resistant**

It may be possible that within 10 years there will be computers that can break current classical public keys. EndCryptor uses classical public keys and new quantum attack resistant public keys.

### **Patented technology, state of the art cipher and public keys**

The protocol that provides the features has been patented in USA. The implementation of symmetric encryption and public keys uses publicly available source code developed by the scientists who designed the systems.

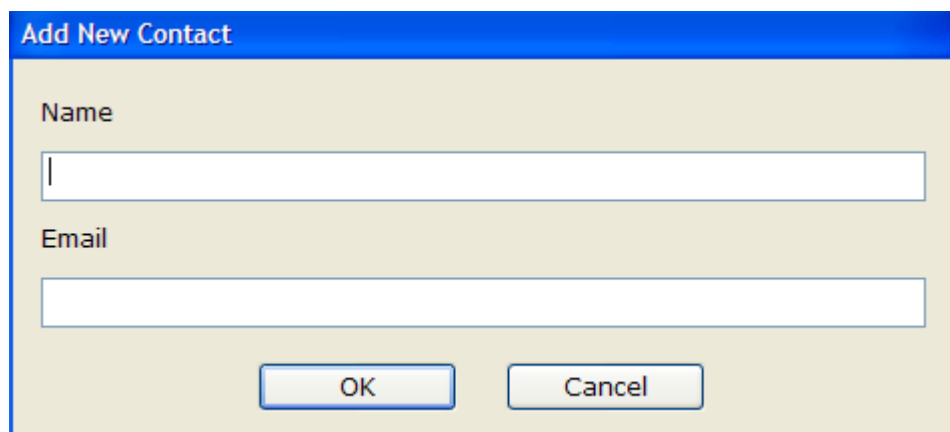
## Quick start guide

Install EndCryptor. After sending one and receiving two verification emails you can start sending encrypted emails to other users of EndCryptor – this is an automated process. They also can send to you.

See the YouTube videos about installing and using EndCryptor:

<https://www.youtube.com/channel/UCAiIkQf2kRmcULg86GIQWRA>

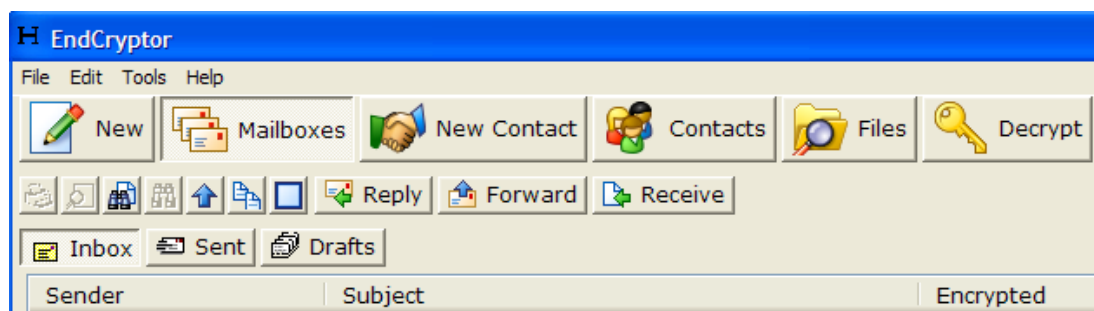
The verification emails check that you are in control of your email address. After verification your email address and long term public key are put into the Web Directory of EndCryptor. If someone wants to start sending encrypted emails to you he/she must know your email address – it is typed into EndCryptor and the Web Directory is searched for the public key associated with the email address.



The image shows a dialog box titled "Add New Contact". It has a blue header bar with the text "Add New Contact". Below the header, there are two text input fields. The first is labeled "Name" and the second is labeled "Email". At the bottom of the dialog, there are two buttons: "OK" and "Cancel".

It is not mandatory to use the Web Directory. In this case the user must know the public key of a new contact.

Main window when new encrypted email has arrived:



## Useful tips

Below are example email settings.

Edit Email Account ✕

Email

Settings for sending

User

Hide character values:  +

Password

SMTP Server

Settings for receiving

Same user and password as in sending

User

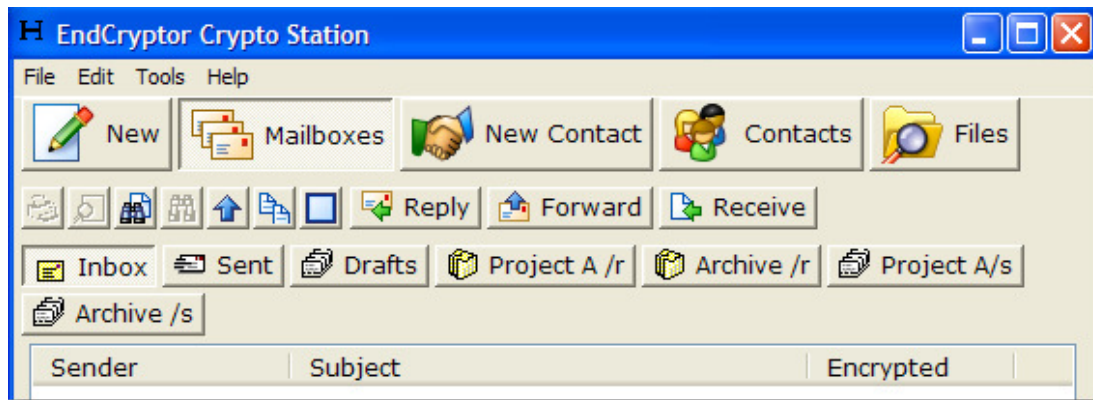
Hide character values  +

Password

IMAP Server

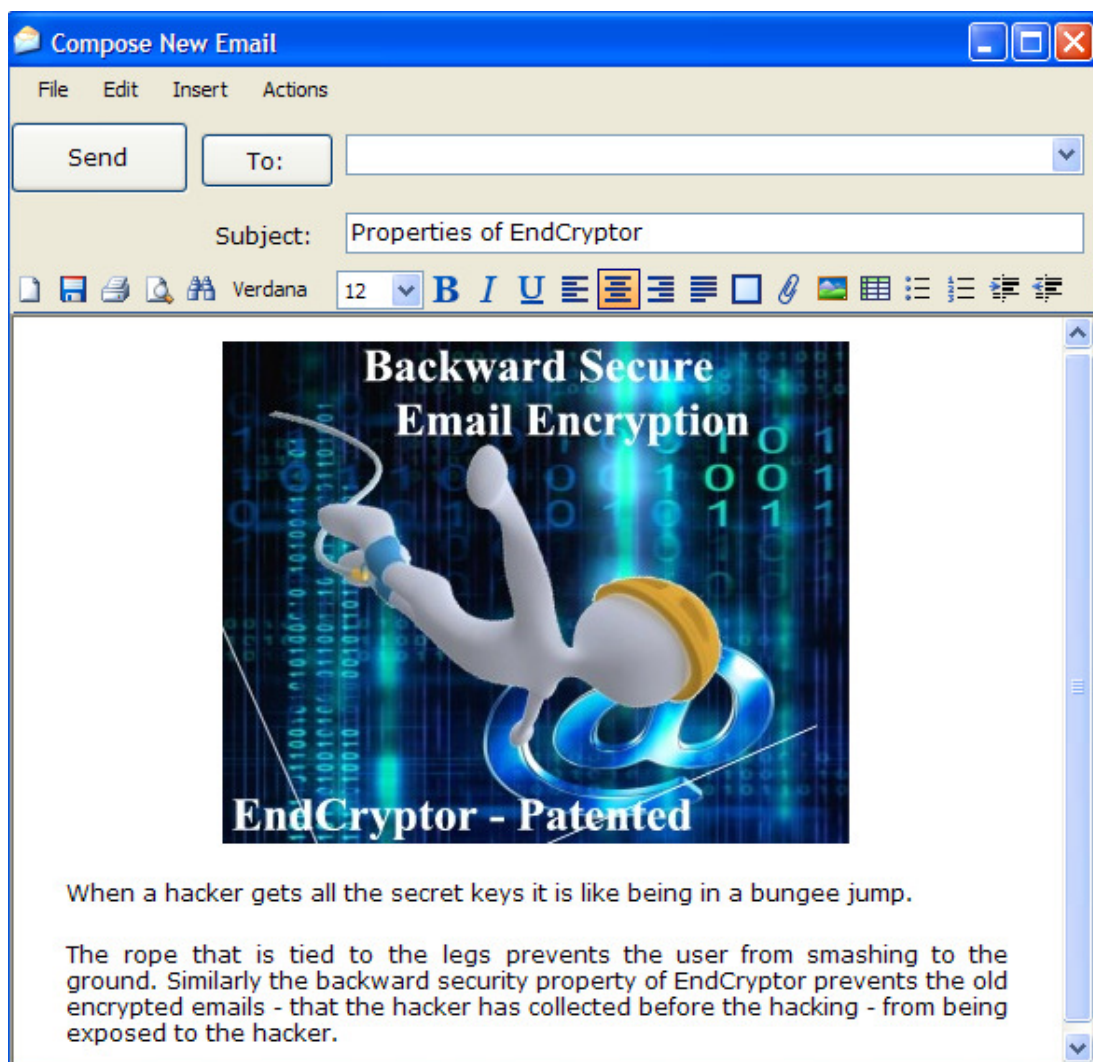
Check for new mail automatically

Create new mailboxes e.g. one named 'Archive /r' which contains processed received emails. It might be convenient to create temporary project based mailboxes whose contents can be moved to 'Archive /r' and 'Archive /s' mailboxes when the specific project has been finished.



New mailboxes are created by placing the mouse over a tab and using the right mouse click.

Example of email composing window:



## Security features explained

EndCryptor protects against attacks done by current classical and future quantum computers. Scientists consider that it may be possible that such quantum computers could be built within 10 years that could break current classical public keys. Therefore cryptographers are developing new kind of public keys - which currently are understood to resist quantum attacks. There are several possible solutions. The quantum attack resistant public keys used in EndCryptor are called supersingular isogeny Diffie-Hellman keys. The reader should keep in mind that encrypted communication can easily be stored and if quantum computers become reality they can be used to decrypt stored old communication.

EndCryptor offers features that are essential for real world protection: **backward security** and **recovery from an attack**. It is important that there is protection when a hacker gets access to current secret encryption/decryption keys.

<i>Comparison between EndCryptor and the S/MIME and the PGP-family of email encryption products (PGP, OpenPGP, GnuPG,...) in case of a successful spying attack which reveals current secret keys - like private keys of public keys - to the attacker</i>		
	<b>EndCryptor</b>	<b>S/MIME and PGP -family</b>
<b>Backward security (= are encrypted messages sent to the victim before the attack protected?)</b>	YES	NO
<b>Recovery from the attack will happen</b>	When the next message from the victim is decrypted. In quantum attack when next quantum attack resistant Diffie-Hellman key exchange is done.	When the new public key of the victim is received. This usually happens at predetermined intervals - after several months or years. No protection against quantum attacks.
<b>Identity theft will be revealed</b>	YES	NO

To enable fast recovery from hacking EndCryptor uses a lot of classical public keys. Quantum attack resistant public keys are used more seldom – they are much slower to use. In typical usage quantum attack resistant Diffie-Hellman key exchange is done at least once a week.

Recently this kind of hacking attack has been done e.g. by malwares **Sauron**, **APT30**, **Red October**, **TeamSpy** and **Mask** - which operated undetected about 5, 10, 5, 10 and 7 years, respectively - and stole among other things encryption keys.

The main targets of e.g. Mask fall into the following categories: government institutions, diplomatic / embassies, energy, oil and gas companies, research, private equity firms, activists.<sup>1</sup>

Without **backward security**<sup>2</sup> and **recovery from attack** a single successful spying attack into your computer leads to the exposure of all previous and future encrypted communication sent **to** you! In some solutions also all communication sent **from** you is exposed – this happens if the solution is such that the sender of a message can decrypt it after its encryption! After a successful attack the adversary does not need to access your computer anymore. What the adversary then needs is encrypted messages created before and after the attack. Using the information provided via the attack they can be decrypted. In the light of recent leaks about state level data interception and collection it is known that encrypted messages (emails, chats ...) are routinely collected and stored.

The spying attack can e.g. be the utilization of dedicated spyware, worm, virus or the usage of a newly published security hole through which the computer can be accessed from the network and then using a keylogger to capture the entry password to the encryption software's database (S/MIME certificate, keyring or whatever it is called) and the password's and the data's transmittal to the attacker. This exposure of the security data can happen other ways also: the user turns **from friend to foe** and reveals his own security data to the adversary; or is **forced** (e.g. by a court order) or **lured** to reveal current security data; etc.

After the exposure old and new **encrypted messages** sent to (from) the victim **can be decrypted** unless the software is prepared to face the exposure of its security database.

---

<sup>1</sup> On August 2016 security companies Kaspersky and Symantec revealed a spying operation named as Project Sauron or Remsec which had run undetected about 5 years. The operation according to Kaspersky was: "designed to enable long-term cyber-espionage campaigns" and "has high interest in communication encryption software widely used by targeted governmental organizations. It steals encryption keys, configuration files, and IP addresses of the key infrastructure servers related to the software." Symantec says about the malware that there is a "module that contains a string named "Sauron" in its code. Given its capabilities, it is possible the attackers have nicknamed the module after the all-seeing villain in Lord of the Rings." On April 2015 security company FireEye reported that malware named APT30 had been found to have been spying 10 years mainly in South East Asia. Among data it collected were files ending with .pgp. On July 2014 F-Secure reported about CosmicDuke malware which had attacked against NATO and European government agencies. This malware stole among other things certificates and their private keys. On February 2014 Kaspersky Lab announced that they had found and analyzed Mask - "an advanced threat actor that has been involved in cyber-espionage operations since at least 2007 ... one of the most complex APT we observed ... more than 380 unique victims in 31 countries ... could be a nation-state sponsored campaign ..." The Red October malware which was also found and analyzed by Kaspersky Lab (results published in January 2013) collected \*.crt, \*.cer (these are certificate related), \*.pgp, \*.gpg, pubring.\*, secring.\* (PGP, and GPG related) files and recorded key presses and values in password fields. The Red October was operating about 5 years. A report published on March 2013 from CrySys Lab in Hungary says about TeamSpy malware: "Many of the victims appear to be ordinary users, but some of the victims are high profile industrial, research, or diplomatic targets". First example of a virus that stole PGP's security database 'keyring' was Caligula virus (1999), this attack did not use a keylogger, but was a proof of concept attack.

<sup>2</sup> In online communication (e.g. chatting, SSL or https) the corresponding term for **backward security** is **Perfect Forward Secrecy (PFS)** – which means that if a message is decrypted securely now it cannot be decrypted again in the future by opponent even if the opponent obtains the encryption keys of that future time.

If **recovery from attack** is provided then after the recovery the attacker must be able to obtain the security data again in order to be able to continue decrypting new messages - this may, however, now be impossible e.g. if the program containing the security hole has been fixed by installation of a proper update.

EndCryptor is a solution that considers the unwanted but realistic possibility that at some point in time the security data - private keys, etc. - are revealed to an adversary. Our results in case of a classical attack: old sent and received communication of the victim is protected and also future sent communication from the victim is protected. The restoration of total security happens when the next message from the victim has been decrypted.

### Detailed Features

- Both the sender and the receiver must have EndCryptor installed. An email account on email server is needed - same account (i.e. user's current email account) can be used for unencrypted emails and encrypted emails. Encrypted emails are typed using EndCryptor and they are sent and received using EndCryptor. A encrypted email is a file that is an attachment in an ordinary email. The sending and receiving is enabled by defining user's email account's SMTP and IMAP settings into EndCryptor, e.g. Gmail can be used.
- The solution is a true decentralized end-to-end encryption solution. Users can change their email addresses and email service providers and the encryption still works - of course contacts must be informed of the change of an email address. Thus there is no central server of Enternet Oy for all users which is needed for email delivery (which could be attacked by hostile parties, nor is there any javascript code that is delivered to users by a central server when using the product, nor is there any server that stores the private keys of users' public keys). This approach also means that Enternet Oy cannot be fooled/forced to deliver hostile code to specific users and that Enternet Oy is not able to decrypt or monitor the email traffic of users.
- The cipher used is 256-bit keysize ChaCha20.
- Encryption keys of messages are determined using **elliptic curve public key technology** (classical: Edwards curve Ed25519 and corresponding Curve25519, quantum attack resistant: SIDH 2.0 and 3.0 supersingular isogeny Diffie-Hellman keys designed by Microsoft).
- At the beginning of the email exchange the user published long term public keys are responsible for the protection of the email. **EndCryptor puts inside the first encrypted emails newly created short term public keys that initialize the patented protocol that continuously exchanges internal short term public keys when emails are being exchanged.**
- Each message ends with an authentication mac and signature. This ensures to the receiver that the message was created by the claimed sender and that the file was not altered during traversal.
- After the decryption the correctness of the plaintext is verified using Poly1305 authentication code.
- The sent and received messages are stored in encrypted form on a user's computer – the user can view their decrypted contents when correct entry



- password to EndCryptor has been given. The stored messages can be searched and moved between different user creatable mailboxes.
- **Messages can be exported** in eml format. They can be imported into email archiving solutions. The exported files are digitally signed to detect tampering. They can also be viewed by many email client programs or dragged and dropped into an existing local email folder (e.g. into Mozilla Thunderbird). The export feature allows the user to have a complete cleartext archive of the communication.
  - The stored messages can be backed up by copying and the **backups can be decrypted using a personal or a companywide (optional) export key**. EndCryptor can take a backup of the security database and restore it. That backup can be encrypted. The stored emails can also be backed up by EndCryptor immediately after they have been written to disk.
  - **Properties under classical attack when the security database of user Alice is exposed by hacker:**
    - Old and future encrypted messages sent from Alice are protected.
    - Backward security: encrypted messages that have been decrypted by Alice are protected.
    - Recovery from an attack: when the next new message from Alice to Bob has been decrypted then the messages from Bob to Alice cannot anymore be decrypted by adversary.
    - Certain kind of protection against identity theft: either the theft attempt fails or it succeeds but then all future messages exchanged between the fooled party and Alice will be rejected. Protection against identity theft is important since a user may have blind reliance on the protection given by a digital signature. If the security data is exposed to a hacker then identity theft can be tried.
  - Reports messages that have not been decrypted. The sender can be sure that the receiver has decrypted the message. Important e.g. when the message contains some latest technical document that must be used by the receiver.
  - **Possibility to delete the keys of missing messages** - if a message is encrypted but not received then the receiver can delete its decryption keys. This requires that the receiver has received a newer message from the sender.
  - **Protection against a replay attack** where an adversary intercepts and copies an encrypted message and later resends it: 1) a message can be decrypted only once 2) the decryption keys of missing messages can be deleted.
  - If EndCryptor is used for sending or receiving it stores the received certificates from the email server. Certificates can be imported and exported to/from the collection of certificates. It can be specified which certificates are allowed to be received – if a new certificate is received the user is prompted for acceptance. This is a highly advanced option and is motivated by the so-called **“compelled certificate creation attack”** or a hacking attack against a Certificate Authority. In those attacks some Certificate Authority has written a certificate of the email server to a wrong party or a hacker has gained the ability to write certificates in the name of the Certificate Authority. Note that some Certificate Authorities have stopped their business because of a successful hacking attack. The possible forgery of certificates is very annoying because the whole idea of certificates is that they can be trusted. If the above

mentioned attacks succeed the already encrypted EndCryptor message that is an attachment in the email stays protected but the attacker gains user's username and password to the email server. For more details read the 'The risks of SSL' part of this document.

- Compression of plaintext. Required amount of random bytes are added to hide the length of this compressed plaintext - encrypted files have different sizes even if their decrypted content is the same. Selected files from the Canterbury Corpus:

File	Size	EndCryptor	bpc
e.coli	4,638,690	1,223,810	2.11
bible.txt	4,047,392	853,556	1.69
world192.txt	2,473,400	474,528	1.53
kennedy.xls	1,029,744	130,285	1.01

bpc = bits per character (byte). EndCryptor was used with the default settings..

- A message may have more than one receiver. Contacts can be grouped.
- File wiping, calculation of a cryptographic hash value (checksum) of a file.
- If an Internet connection is considered to be too risky then EndCryptor **can be run entirely disconnected from the network**. When a message is encrypted a list of its receivers can be stored in a text format, the message and the list of its recipients can be stored in user given folder. The encrypted message and this list are moved to the actual sending machine using removable media. When decryption is needed the encrypted message is delivered to the receiving EndCryptor again using removable media. EndCryptor can be set to monitor some user given folder for new encrypted messages. A custom made program can be defined so that it is used whenever a message is being sent.
- The security database and the stored sent and received messages can be moved to removable media and accessed from it. Thus it is possible to use EndCryptor both from office and laptop computers. The size of an empty database is about 1 MB.

## Tutorial on public key technology

Public key technology is the basis of modern protected communication. This short tutorial explains briefly without technical details the most important things to know about public keys. We also explain briefly some of our protection mechanisms against the known attack points of public key based systems.

We use public keys for these reasons:

- To form a shared secret
- To recover from attack
- To form a digital signature

Main attack types:

- Stealing of a private key
- Man-in-the-middle during key exchange

### **Public keys enable the formation of a shared secret.**

When two persons exchange public keys which they have created they can calculate a value that only they know. A third person that sees the public keys exchanged cannot calculate this value. The calculated value is called a shared secret. It is typically used later as an encryption key to encrypt the communication between the parties. This method is called Diffie-Hellman key exchange according to its inventors Whitfield Diffie and Martin Hellman.

This solves a very important problem: how to communicate securely an encryption key to the other person? By sending and receiving a public key.

Each public key has a corresponding private key. The creator of the public key automatically knows this private key. The shared secret is calculated by the help of this private key and the other person's public key.

### **Public keys enable the recovery from attack.**

Now the third person that watches the exchange of public keys cannot calculate the shared secret that the creators of the public keys can calculate. However, if he successfully sends a spy program and **steals a private key** from one of the parties then the shared secret becomes known to him and he can decrypt messages created after this public key exchange.

We have now a new problem: how to recover from this spying attack? EndCrytor solves this by creating new public keys and sending them.

The attacker must again be able to steal a private key – if he cannot do this he cannot anymore decrypt new messages.

Some public key based systems use a same public key for years. If its private key becomes available to an adversary e.g. via hacking all communication under shared

secrets calculated from this key become known to the adversary. Computer viruses that search for private keys are known to exist.

EndCryptor creates a lot of public keys. Each encrypted message after the first messages that use the long term public keys contain new short term public keys of the sender. These public keys are specific to the receiver in question. When a person whose private key has been stolen sends a new message and when it is received by the other party then a new shared secret can be calculated – the attacker has lost his ability to decrypt messages sent to the victim.

There is still another problem: how to protect old messages received prior to the attack?

Please note that a person may have received several messages without sending new messages and then the stealing of the private key happens. How to protect these messages received between a Diffie-Hellman key exchange and an attack? The answer is a bit complicated and we give here only the result:

Using our patented solution those encrypted messages that the attacker has captured and the proper receiver decrypted prior to the attack cannot be decrypted by the attacker.

More information about our solution can be found on cryptographic technical details page.

### **Public keys enable the formation of a digital signature.**

Each message has a digital signature as the last part of the message. The signature is formed by first calculating the cryptographic hash value (checksum or digest) of the actual message and then with the help of a private key the digital signature is calculated and appended to the end of the message.

The person who receives the message and the signature then verifies the signature with a public key and by calculating the cryptographic hash value of the message.

If the message or the signature itself has been modified by an attacker during traversal in the net then the signature will not verify – only the person who has the private key can create proper signatures which will verify correctly only by the corresponding public key.

This solves the problem: how to prevent the modification of messages and the falsification of the sender's identity?

### **Man-in-the-middle attack**

This attack can happen if a person sends a public key to another person. A third person, an attacker Mallory can replace the sent public key with the public key he has created:

Alice sends a public key to Bob but Mallory intercepts it and creates his own public key and sends it to Bob. Bob creates his reply that has his public key and sends it to Alice but again Mallory intercepts the message and the public key and creates another public key and sends it to Alice. Now Mallory can impersonate both parties.

Man-in-the-middle attack: Alice  $\leftarrow \rightarrow$  Mallory  $\leftarrow \rightarrow$  Bob.

Alice and Bob do not know that there is Mallory between their communication who replaced their public keys with the public keys created by Mallory.

To protect against this attack EndCryptor's Web Directory stores user's long term public key and email address. When the email address verification has been done the public key and the email address are signed by a public key which has a signature chain to public key that EndCryptor knows. When a user's long term public key is fetched from the Web Directory or an encrypted email contains sender's long term public key this signature is checked by EndCryptor. A user can check that the values in the Web Directory correspond to his/her email address and public key.

EndCryptor provides a method to reveal a man-in-the-middle attack after some messages have been exchanged: e.g. telephone conversation can be done to compare checksums. Also checksums at the start of the email exchange are stored into the database on user's computer and can be viewed any time later.

## **EndCryptor, S/MIME and PGP under attack**

We study here the exposure of a private key under attack by classical computer.

### **Exposure of a private key**

The reader should recall that the exposure of a private key to an adversary exposes all communication that uses the shared secret calculated with this private key. In practice this means that in PGP and in S/MIME all communication sent to the victim are revealed to the adversary.

The S/MIME email encryption method uses a public key infrastructure (pki) which means that there is a Certificate Authority that digitally signs every new public key. Users already have the public key of the Certificate Authority and use this public key to verify the signature of a certificate that contains the new public key of a user. When a new public key is introduced it must first be certified by a Certificate Authority and then delivered in a certificate to a user.

In S/MIME and PGP the public keys are changed usually at intervals of years.

In EndCryptor at the beginning of the email exchange the user published long term public keys are responsible for the protection of the email. EndCryptor puts inside the first encrypted emails new public keys that initialize the patented protocol that continuously changes internal short term public keys when emails are being exchanged.

Suppose now that Alice starts communicating with Bob using EndCryptor and sends an encrypted email to Bob using Bob's long term public key. After receiving the email Bob replies to it. Later an adversary finds out Bob's long term private key. In EndCryptor only the first Alice's email to Bob can be decrypted by the adversary whereas the traditional systems expose all Alice's later emails to Bob which were sent to Bob's public key.

Using our method the short term public keys are changed more frequently: if the parties communicate in turns one public key is used only once. An exposed private key has a very short life time in our solution. Our solution for the renewal of short term public keys is cost free.
--

## The risks of SSL

This chapter is about the risks of relying on TLS/SSL encryption - which is currently the only universal encryption protocol supported by all web browsers when connecting to websites (the web browser typically displays then a lock on the address bar - trying to convince the user of the security of the connection - and may also show the protocol name 'https').

On March 2017 WikiLeaks published leaks from the hacking arsenal of the CIA (USA's Central Intelligence Agency). In some of those documents there are advices to malware writers: **'DO NOT solely rely on SSL/TLS to secure data in transit. Numerous man-in-middle attack vectors and publicly disclosed flaws in the protocol.'** and **'because this outer layer may be decrypted by an attacker (e.g., SSL Man-in-the-Middle) any transport encryption must be used for traffic blending only and not for secrecy.'**

Previously on November 2011 the Wall Street Journal published the 'Surveillance Catalog' and the WikiLeaks organization provided a list of International surveillance companies and their equipment on the 'WikiLeaks Spy Files' publication. Some examples from the brochures that describe the properties of the equipments: **"It can also decrypt SSL traffic if installed in MITM (man-in-the-middle) configuration ..."**; **"Track the suspect's encrypted communication using Gmail, Hush mail etc., Track the suspects banking transactions etc."**; **"Intercept any communication within Secure Socket Layer (SSL) or Transport Layer Security (TLS) sessions. Once in place, devices have the capability to become a go-between for any TLS or SSL connections ... users are lulled into a false sense of security afforded by web, e-mail or VoIP encryption."**; **"But with a 'man in the middle,' the ... technology is able to intercept the traffic and the certificate and send along its own fake certificate to the computer, making the computer think traffic is flowing normally."** Read below a detailed explanation of how this is possible.

When a user connects to a HTTPS/ (SSL or TLS) server, the server sends a certificate to the user which ensures to the user that he really is connecting to the wanted server. How can a certificate do that? The owner of the server has – before starting his services - contacted a Certificate Authority (CA) and proved to him that he owns and controls the server. The owner of the server has sent a public key of the server to the CA and the CA has signed this public key using the private key of the CA. When a user receives the certificate his web browser checks that the CA's signature is valid using the stored public keys of the well known CA. There are about 600 CAs and current web browsers store their public keys and also update them if that is needed. When the CA's signature has been checked then the user's browser checks that the data coming from the server has a valid signature which is signed by the public key of the server (which is in the certificate).

Note that currently any CA can issue a certificate for any website. If the CA decides so it can write a certificate for any website and can use any public key as the public key of the server – this is against the rules but no one can prevent the CA from actually doing this. It may also happen that no one notices these actions – certificates are not normally shown neither are they stored for later inspection. An improvement

to this situation is the Certificate Transparency project started by Google which is explained in more detail later in this chapter.

There is special equipment available that is placed in the middle of the communication between the user/victim and the server. These devices are designed to use also intermediate level CA certificates – they can generate the needed certificates as a need arises<sup>1</sup>.

---

<sup>1</sup> Certificate Authority Trustwave admitted on February 4, 2012 that they had given one private customer an intermediate certificate authority certificate inside a special machine which generated certificates for any website. This was done to decipher and monitor all company's online SSL/TLS communication regardless whether the devices used were company provided or not – because the certificate was issued by a Certificate Authority no new certificates were needed in users' computers.

On January 3, 2013 Google reported that they had on December 24, 2012 detected an unauthorized digital certificate for the "\*.google.com" domain. The certificate was issued by an intermediate certificate authority linking back to TURKTRUST, a Turkish certificate authority. Intermediate CA certificates carry the full authority of the CA, so anyone who has one can use it to create a certificate for any website they wish to impersonate. See Google's blog entry, ([googleonlinesecurity.blogspot.com/2013/01/enhancing-digital-certificate-security.html](http://googleonlinesecurity.blogspot.com/2013/01/enhancing-digital-certificate-security.html)).

TURKTRUST told Google that in August 2011, they had mistakenly issued two intermediate CA certificates to organizations that should have instead received regular SSL certificates. Please note that this kind of certificate is exactly that kind of certificate that can be used in the man-in-the-middle machines to monitor any intercepted traffic to any website, the fake certificates generated by this intermediate certificate may have been used during about 16 months.



Following are certificate related attacks:

1. CA (established for the purposes of intelligence gathering for a country A's intelligence agency) issues a certificate for a server in a country B to a public key of this intelligence agency.
2. CA has been hacked. The attacker has obtained the private key of the CA and can issue certificates which the user's web browser decides to be valid<sup>1</sup>.

---

<sup>1</sup> Certificate Authorities can be targeted by viruses, e.g. Duqu targeted certificate authorities and used stolen and forged certificates for its purposes. Electronic Frontier Foundation's SSL Observatory project report (2011-10-27, <https://www.eff.org/deeplinks/2011/10/how-secure-https-today>) that the following reasons for certificate revocations were found in Certificate Revocation Lists:

reason	occurrences
NULL	921683
Affiliation Changed	41438
CA Compromise	248
Certificate Hold	80371
Cessation Of Operation	690905
Key Compromise	73345
Privilege Withdrawn	4622
Superseded	81021
Unspecified	168993

The researchers say (2011-10-27) that: "In at least 248 cases, a CA chose to indicate that it had been compromised as a reason for revoking a cert. Such statements have been issued by 14 distinct CA organizations." When the statistics from earlier 4 months are compared to above findings: "So, from this data, we can observe that at least 4 CAs have experienced or discovered compromise incidents in the past four months. Again, each of these incidents could have broken the security of **any HTTPS website.**"

3. CA has been forced (by an order from the country's authorities) to issue a certificate for the public key of the attacker (law enforcement). This is called '**compelled certificate creation attack**'<sup>1</sup>.
4. The private key of the SSL server has been exposed. If the server has not been configured to use **Perfect Forward Secrecy** (PFS) the recorded old SSL sessions can be decrypted. If PFS is used a man-in-the-middle attack is required at session time for decryption of the traffic. The attack is now easier to do since no additional fake certificate is needed since server's private key is known<sup>2</sup>. The **Heartbleed vulnerability** in OpenSSL that was found in April 2014 exposed server's memory (private keys etc.). The bug was undetected in the code for 2 years but even older recorded SLL sessions (without PFS) can be opened using an exposed private key<sup>3</sup>.

---

<sup>1</sup> The term 'compelled certificate creation attack' was introduced by Christopher Soghoian and Sid Stamm in their paper 'Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL', in Financial Cryptography and Data Security '11 March 2011.

On December 2013 Google noticed that several unauthorized certificates were issued for Google's domains. The certificates were issued by a French governmental certificate authority ANSSI who said that the issuing of the certificates was a human error.

On July 8, 2014 Google reported (<https://security.googleblog.com/2014/07/maintaining-digital-certificate-security.html>) that they had found fake certificates issued for several Google domains and one Yahoo domain and maybe for some other domains also. The issuer of the certificates was India's National Informatics Centre. India's Controller of Certifying Authorities said that the issuer's issuance policies were compromised.

On March 23, 2015 Google reported (<https://security.googleblog.com/2015/03/maintaining-digital-certificate-security.html>) that an intermediate certificate authority based in Egypt had used an intermediate level certificate in a proxy to create certificates for user's SSL sessions. The used intermediate level certificate was issued by Chinese certification authority CNNIC.

<sup>2</sup> Recent revelations of state level spying have emphasized the importance of PFS and some big service providers have started to use it. **Note that PFS is just that what EndCryptor provides in email encryption: future attacks can't expose old traffic.**

<sup>3</sup> See [www.heartbleed.com](http://www.heartbleed.com) , "We attacked ourselves from outside, without leaving a trace. Without using any privileged information or credentials we were able steal from ourselves the secret keys used for our X.509 certificates, user names and passwords, instant messages, emails and business critical documents and communication."

5. The attacker uses vulnerability in some software and then installs the attacker created certificate into a trusted certificate store on victim's computer - this enables the attacker to perform man-in-the-middle attack on victim's SSL/TLS web browsing sessions. The attacker needs no software on victim's machine - the installed certificate enables the attack<sup>1</sup>.

In the abovementioned attacks 1-3 the attacker must be able to mimic the real server and/or do a man in the middle attack where he gets the data from the user and sends it to the real server and also sends the server's response back to the user. The attack allows the attacker to see and modify user's traffic to/from the server in unencrypted form. **Note that in these attacks 1-3 the attacker does not need access to user's computer or to the server.** One has to consider also the possibility that also non law enforcement parties may have obtained the equipment for the man in the middle handling and can use it in the attacks. The attack number 1 is challenging because the traffic needs to be routed via another country, it is however possible to change the routing tables of Internet or hacked routers to achieve this.

**The SSL attack can be tried on 'normal' SSL or TLS based email and webmail solutions and on email encryption solutions that are web-based. There are also Virtual Private Network solutions that use the web browser and SSL. The attack on these systems can be tried always when the SSL connection is done. Web based systems usually use marketing argument that no software is needed on user's computer because only a web browser is needed.**

**EndCryptor encrypts the message before contacting an email server; even a successful SSL attack cannot expose the message.** In case of EndCryptor the attacker thus can only gain the userid and password to the email server. EndCryptor also stores every certificate it receives, they can later be analyzed if an SSL attack is suspected. EndCryptor can be configured so that when it connects to an email server using SSL it accepts only certain already received certificates – this prevents the attack, the dishonest certificate has not been seen before and is rejected. This technique is called certificate pinning.

There are also devices that do SSL DPI (SSL Deep Packet Inspection, another term used is 'SSL bridging') inside a specific company using the man-in-the-middle method to decrypt SSL traffic flowing in and out of the company. Also some firewalls, antivirus and parental control programs can be configured so that they decrypt and re-encrypt the SSL traffic in order to examine the decrypted traffic. In these settings an intermediate level certificate is placed into the man-in-the-middle device or software and its root certificate is placed into company's computers<sup>2</sup> - the intermediate level certificate and its root are self-signed by the company in question and thus only this company's traffic can be monitored without users noticing anything. If the user's computer does not contain company's certificate then user's web browser issues a warning – which the user, however, may choose to bypass (this

---

<sup>1</sup> This technique is mentioned in the leaked material of Hacking Team company that sells spyware to governments and law enforcement agencies who can easily perform the MITM attack by compelling the internet service providers to place the MITM machines at proper places.

<sup>2</sup> The Web Debugging Proxy Fiddler uses the same technique to log all HTTPS traffic between a computer and the Internet. Another tool is SSLsniff which is designed to MITM all SSL connections on a LAN, and dynamically generates certificates for the domains that are being accessed on the fly.

depends on the browser and its settings)<sup>1</sup>. On mobile devices certain browsers (Nokia's Xpress Browser on old Nokia devices and Opera Mini browser) use man-in-the-middle technique to decrypt and re-encrypt SSL/TLS traffic in a proxy server, the motivation is to compress data and lessen the computing resources needed on the mobile device.

The Certificate Transparency project by Google tries to improve the certification infrastructure. According to <https://www.certificate-transparency.org/benefits> : "Indeed, incidents that at one time were concealed and downplayed, and in fact caused the shutdown of an entire CA, could be exposed much earlier and mitigated by simply revoking a single certificate."

This project tries to log all CA issued SSL/TLS certificates in the world, major CAs take part of it and also search engines may submit certificates they see into the logs. Certificates issued by a publicly accepted CA after April 30, 2018 will not be accepted as secure by the Chrome browser unless they have a signed statement (a Certificate Transparency extension embedded into the certificate) that the certificate will be logged.

An owner of a domain (e.g. example.com) can query from the logs all the certificates issued to a domain and check that there are only proper ones. The logging of certificates is not done to local certificates that are not created by a publicly accepted CA (Certification Authority) and that are added to the certificate store of user's computer by the user or by some program like antivirus, firewall and parental control program or malware.

Encryption solutions relying on SSL/TLS when communicating to servers do not necessarily follow the same practice as browsers i.e. require proper Certificate Transparency extension in the certificate.

---

<sup>1</sup> Citizen Lab's report 'Planet Blue Coat: Mapping Global Censorship and Surveillance Tools' (<https://citizenlab.org/2013/01/planet-blue-coat-mapping-global-censorship-and-surveillance-tools/>) describes how SSL interception machines intended for legitimate use for monitoring a specific company's traffic are also used by countries with a history of concerns over human rights.

## Cryptographic technical details

Both parties that send and receive emails need that EndCryptor is installed on users' computers in order to encrypt and decrypt. There is no need to place any code on user's email servers.

The encrypted email is a file that can be delivered by any means available from the sender to the receiver. Typically this is done via the users' email systems. EndCryptor uses email standard's IMAP commands to receive the file that is an attachment in an ordinary email. Same email account can be used for ordinary unencrypted email and for encrypted email.

When a new contact is being added then its long term public key can be fetched from the Web Directory if wanted - this public key is signed by a signature chain that the program code trusts.

The solution is a true decentralized end-to-end encryption solution. Users can change their email addresses and email service providers and the encryption still works - of course contacts must be informed of the change of an email address. Thus there is no central server of Enternet Oy for all users which is needed for email delivery (which could be attacked by hostile parties, nor is there any javascript code that is delivered to users by a central server when using the product, nor is there any server that stores the private keys of users' public keys). This approach also means that Enternet Oy cannot be fooled/forced to deliver hostile code to specific users and that Enternet Oy is not able to decrypt or monitor the email traffic of users.

When encrypting/decrypting the stored information on the EndCryptor's security database on the used computer is used together with the information that the message in question provides. The security database is encrypted, some parts with AES, others with ChaCha20, key size is 256 bits. User's entry password to the security database is hashed using salted password hashing pbkdf2 with hmac sha256 using 10000 iterations.

Used classical elliptic curves in emails are the Edwards curve Ed25519 and the corresponding Curve25519. The Edwards curve is used for signing and the Curve25519 for Diffie-Hellman calculation. The classical security of Curve25519 is 128 bits.

Quantum attack resistant public keys are supersingular isogeny SIDH (p751) 2.0 and 3.0 keys designed by Microsoft. The classical security of a SIDH key is 192 bits. In scientific papers authors of SIDH (p751) construction state that its quantum security is 128 bits, in the terminology of NIST's Post-Quantum Cryptography project it is "matching the post-quantum security of AES192" - this refers to NIST's Quantum Security Strength Categories III. In January 2019 SIKE which has SIDH as its core part was accepted to Round 2 in NIST's Post-Quantum Cryptography Standardization Process.

Compare classical cryptographical strengths:

<b>Symmetric</b>	<b>Elliptic (classical)</b>	<b>DH or RSA</b>
80	163-223	1024
112	224-255	2048
128	256-383	3072
192	384-511	7680
256	512+	15360

The classical security of curve25519 is 128 bits and matches that of a 3072 bit RSA/Diffie-Hellman public key. Note that the classical security of 192 symmetric bits corresponds to 7680 DH or RSA public key bits. The reader should note that usually cryptographic construction's security is expressed as the security of its weakest link – this is usually the public key.

This table appears in NIST Special Publication 800-57 from July 2012 titled 'Recommendation for Key Management – Part 1: General(Revision 3)'. NIST means National Institute of Standards and Technology (USA).

If the parties communicate in turns then the first email's classical security is 128 bits, after that the classical security is 192 bits. The quantum protection starts from the second email (included). The patented protocol starts after the second email has been received.

The implementation of the Ed25519, Curve25519, Chacha20 and Poly1305 is the reference source code implementation available from SUPERCOP benchmark suite and NaCl crypto library (European Network of Excellence in Cryptology II projects funded by European Commission). These primitives are designed to give protection against side channel attacks like cache timing attacks. The implementation of the SIDH 2.0 and 3.0 public keys is from GitHub: PQCrypto-SIDH. The code is also constant time code.

The signatures in encrypted messages use Keccak-256 which is constructed according to the specification that won the SHA3 competition (bit rate is 1088 and capacity is 512).

The private keys of public keys are made using a Goldreich-Levin hard-core bit generator. The seed to the generator consists of events like mouse movements and their timings and bytes provided by the BCryptGenRandom system call.

The first messages encrypted using long term public keys consist of classical part and SIDH part. The SIDH part is appended to the end of the classical part. If the parties communicate in turns there are 2 messages of this kind.

EndCryptor's security properties rely on the protocol that needs to be initialized with 2 short term (ephemeral) classical public keys created at the time of the contact creation. These keys are included as an encrypted part of the first exchanged emails' classical part.

The first emails' SIDH part contains SIDH 2.0 or 3.0 quantum attack resistant public keys. After the initial exchange the SIDH keys are version 3 keys if both parties have at least version 2.5.4.62. New SIDH keys are later exchanged at intervals of 7 days if parties communicate frequently.

The encryption of a first message to a receiver's long term public key consists of creating an ephemeral Ed25519 public key which is used to calculate a shared secret with receiver's long term public key. Additional ephemeral Ed25519 public key is created and put into that part of the classical part that will be encrypted. The whole classical part is signed with sender's long term classical public key. That part of the message which is encrypted includes the signature at the end of the classical part. The hashed value of shared secret is used as an encryption key for ChaCha20, key size is 256. The last SIDH part of the message contains newly created SIDH keys and a signature of the whole message signed with sender's long term classical public key.

When the above message has been received and a first message to its sender is sent the sender creates an ephemeral Ed25519 public key and new SIDH keys and calculates a shared secret with that Ed25519 key that was inside the received encrypted message.

The actual encryption/decryption key of the second message is calculated by computing a Keccak hash value over a value derived from the classical shared secret and the SIDH shared secret i.e  $\text{key} = \text{hash}(A \parallel B)$  where  $\parallel$  is the concatenation operation, A is the hash of classical shared secret and B is derived from the SIDH shared secret. In more detail: the SIDH shared secret is 192 or 188 (version 3.0) bytes long, it is Sha3-512 hashed. The first 32 bytes of the result are used in this encryption/decryption key computation as value B.

An above described message's classical part contains 4 latest classical public keys of its sender. An older one signs the next newer one.

After the exchange of initial messages the protocol is initialized. Then a shared secret is calculated with each pair of exchanged protocol initialization classical public keys i.e. two shared secrets are calculated. They are concatenated and a Keccak hash is calculated over the above mentioned two classical shared secrets and a value derived from the SIDH shared secret i.e  $\text{initialization value} = \text{hash}(A1 \parallel A2 \parallel B)$ , where A1 and A2 are classical shared secrets and B is derived from the SIDH shared secret. The result is to be used as initialization value to form protocol's initial states. The reader should note that an attacker has to find out every calculated shared secret to find out the resulting Keccak hash value. In more detail: the SIDH shared secret is 192 or 188 (version 3.0) bytes long, it is Sha3-512 hashed. The last 32 bytes of the result are used in this initialization value computation as value B.

The rest of the discussion below considers the situation when the initialization set of public keys has been received by both participants and the protocol is working. The idea of the protocol is to continuously exchange new classical and post quantum public keys to enable fast recovery from hacking attack. The protocol also protects old emails so that they cannot be decrypted by the attacker if they have been previously decrypted by the true receiver.

An encrypted message is signed by a previously delivered classical public key that the receiver is known to have – these public keys in messages are delivered to the receiver in encrypted form; they are encrypted together with the plaintext. The signature is encrypted and the message ends also with a Keccak mac, also the ephemeral public key in the message is encrypted (the encryption key for the ephemeral key and the signature and the key for the Keccak mac is either derived from the initial shared secret or is a hash of a public key that was delivered in encrypted form). A reader may ask why there are two authentication methods: signature and mac. The signature is useful if one of the parties is hacked: an attacker cannot impersonate the unhacked party based on information obtained from the hacked party. The Keccak mac is checked first, then the signature. Note that a message is accepted as original only if the plaintext's authentication code evaluates correctly – see the end of this chapter. Why the additional encryption of the ephemeral public key and the signature? Why not? The less information is given to an observer the better. When EndCryptor was initially released these values were not encrypted – later it was realized that it can be done without affecting the protocol and that there was a value available that could be used as encryption key. One can also argue that a quantum computer cannot break a symmetrically encrypted classical public key.

Encryption of the messages is done using 256 bit key sized ChaCha20.

The plaintext ends with an authentication code, the authenticator is Poly1305 one time authenticator. During encryption both the plaintext and the authentication code are encrypted. After the decryption the authentication code is calculated over the plaintext and checked.

An outline of the methods used:

**Backward security:** Every EndCryptor message is encrypted with different symmetric 256-bit key and after the message has been decrypted there is no information in the security database from which the decryption keys could be deduced again. A message can thus be decrypted only once.

**Recovery from attack:** Every message EndCryptor encrypts contains new classical public keys of the sender that are specific to the receiver; these public keys are created at the time of sending - when the receiver decrypts the message the security is restored. These classical public keys are delivered in encrypted form; they are encrypted together with the plaintext. Quantum attack resistant public keys are also delivered in encrypted form but more seldom – if the parties communicate regularly the exchange happens at 7 day intervals.

**Identity theft** will be revealed even under spying attack: the stored security data that is used to build symmetric key changes after every decryption and depends on the just decrypted message.

The protocol is a stateful protocol. It means that two states are maintained for each contact: one for sending and one for receiving. A new state and the needed symmetric keys for encryption/decryption and plaintext's mac calculation are constructed from the current state and a calculated Diffie-Hellman (DH) shared secret whenever



sending or receiving happens. The calculation of a new state and the symmetric keys is irreversible i.e. one-way and done using a Goldreich-Levin hard-core pseudo random bit generator (PRG): **PRG(state, DH shared secret) -> list of bits**. From the generated list of bits a new state and the needed symmetric keys are separated. The generated bits pass the next-bit test – it is infeasible to predict bit  $i+1$  if the first  $i$  bits are known. The irreversibility of the construction and the next bit property cause the backward security property of the protocol. The recovery property is caused by using new public keys in messages – which affect the DH shared secret parameter of the PRG.

### **The produced new state and the symmetric keys from the construction:**

**PRG(state, DH shared secret) -> new state and symmetric keys**

**will be unknown to the attacker if the attacker knows only one but not both parameters of the PRG. The state consists of more than 256 bits and it is stored in encrypted form on user's computer.**

If there is available a quantum attack resistant DH result then the actual DH shared secret used in above and below formulas is a Sha3-256 hash over the classical and quantum attack resistant DH shared secrets.

If the attacker has accessed user's computer and has found out all data on the encrypted security database and then loses access to the computer he/she will know the classical public key of an outgoing message but has to break it (or its DH counterpart which the attacker now knows) in order to decrypt the message. When the receiver of this message sends next message to the victim the attacker cannot decrypt it without breaking a public key (if the attacker was not able to decrypt the message from the victim of the hack then he/she must break 256 bit symmetric cipher to get the victim's public key and then break one of the public keys).

To understand the PRG construction the reader needs to be skilled in cryptography.

*Hash(x)* uses ChaCha20 to produce 768 bits that form the 256 bit sized blocks  $h_1$ ,  $h_2$  and  $h_3$ .

*State* consists of 256 bit sized blocks  $s_1$ ,  $s_2$  and  $s_3$ .

*GL(r,x)* produces one Goldreich-Levin hard core bit from  $x$  using random bits  $r$ .

$+$  is the xor operation.

PRG(state,DH shared secret) =

Hash(DH shared secret) to produce blocks  $h_1$ ,  $h_2$ ,  $h_3$

$b_1 = s_1 + h_1$

$b_2 = s_2 + h_2$

$b_3 = s_3 + h_3$

For ( $i=0;i<1280;i++$ )

{

$b_1, b_2 = \text{SHA512}(b_1, b_2)$

produce bit  $\text{GL}(b_3, b_2)$

}

The produced list of bits form the next state's blocks s1, s2 and s3 and the required symmetric encryption/decryption and Poly1305 mac keys. The SHA512 calculation is done using NaCl library's reference, constant time implementation. The GL calculation neither indexes arrays nor branches using secret data.

Security professionals wishing to know more about the protocol should consult the **US Patent 7,899,184 B2** titled "ENDS - Messaging protocol that recovers and has backward security".

#### *Description of encryption of storage files*

When an encrypted email is sent or received it is encrypted again for storage on user's computer (using different encryption keys than those in the email that is traversing the internet). Each storage file is encrypted using different ChaCha20 256 bit key.

When EndCryptor is started the first time it saves a Personal Export key file and the user is asked to create that file's password. These items can be used to decrypt and export user's emails from backup media. Additionally a Company Export Key can be used.

The Personal Export key is actually two keys: a public key/private key pair of an Ed25519 curve and a symmetric Chacha20 256 bit key.

An encrypted storage file has a field F which stores in encrypted form the file's actual encryption key so that the file can be decrypted and exported from backup media.

For Personal Export key the encryption of the field F happens followingly:

1. Create an ephemeral public key.
2. Compute Diffie-Hellman shared secret with this ephemeral key and user's Personal Export public key.
3. Use Keccak to hash the shared secret to key K. Encrypt file's encryption key with this key K. Store the encrypted value of F on the storage file and store the hash of user's Personal Export public key on the storage file.
4. Encrypt the used ephemeral public key with a symmetric Chacha20 256 bit sized key and store it on the storage file.

The user's Personal Export key file stores the private key of user's Personal Export public key and the symmetric key that is used to encrypt the ephemeral public key. The private key is not stored on user's security database but the public key and the symmetric key are.

Company Export Key consists of a public/private key pair.

For a Company Export key the encryption of the field F happens followingly:

1. Create an ephemeral public key.
2. Compute Diffie-Hellman shared secret with this ephemeral key and company's Export public key.

3. Use Keccak to hash the shared secret to key K. Encrypt file's encryption key with this key K. Store the encrypted value of F on the storage file and store the hash of Company Export public key on the storage file.
4. Store the ephemeral public key on the storage file.

The company's Export key file stores the private key of company's public key. If a user wants to use a Company Key the user imports Company's public Export key which is used as described above.

In above storage file calculations the Ed25519 curve points are converted to the corresponding Curve25519 points when calculating the Diffie-Hellman shared secret. The password hashing scheme in Export Key files is a salted password hashing pbkdf2 with hmac sha256 using 30000 iterations.

## To: Really security conscious user

A really security conscious reader should notice that the attacker's possibilities increase if he has the possibility and knowledge to *modify* the contents of the security data or the software in participants' computers. He could e.g. try to install his modified copy of the encryption software that behaves like the proper one but delivers to the attacker the required information.

To prevent *software modification* EndCryptor.exe is digitally signed using Microsoft Authenticode, the signer is "Enternet Oy". When the program starts however the Windows loader will not check the signature – this is because the checking may be very time consuming. The user can check the signature by placing the mouse over the file and using the right mouse click to select properties and Digital Signatures tab and then by pressing the Details button. Please note also that if the signature does not verify the program will still run. EndCryptor.exe itself checks that the cryptographic hash values of its own program files are as defined in the program code of EndCryptor.exe. The hash value of EndCryptor.exe (that of the running program from the media where it is started) is compared to a value stored on the security database (protected by user's entry password). If a reinstallation of previous installation is done EndCryptor should not give any program code modification message at startup. Such a message is given if the running code's checksum differs from that of a previous installation. As further protection EndCryptor can be run from read-only media.

Sometimes it is claimed that encryption products prevent antivirus programs to find viruses because the viruses in encrypted attachments are encrypted and thus undetectable. Typically the antivirus programs check a file when it is written on disk and in case of EndCryptor the virus will be found then. To test your antivirus program with EndCryptor use the EICAR Anti-Virus or Anti-Malware test file from the European Expert Group for IT-Security.

Note that the newest or specially targeted viruses are not detected by antivirus programs. Thus the most secure but uncomfortable usage that protects EndCryptor's program code and encrypted security database is such that EndCryptor is used on a machine not connected to any network and if messages contain attachment files the attachment files are never opened/activated on this machine but moved to another machine for reading/editing. In other words the machine containing EndCryptor should be used for encryption/decryption purposes only. There should be one machine connected to outside world via network that sends/receives encrypted messages, the second machine containing EndCryptor and third or more machines possibly in internal network that are used to manipulate (read/edit) received and sent attachment files in messages. The motivation for separating the machine containing EndCryptor also from the internal network is to minimize the possibility of hostile code being run in that machine if an attachment containing hostile code is opened.

## **Avoid security through obscurity**

When you are considering buying a crypto product demand that you get a clear written description of the cryptographic essentials of the product – this can be a number identifying a patent or another written description. The purpose of this is to enable the verification of the claimed cryptographic properties. Also when new crypto attacks become known their effectiveness against the product can be checked via the description. The software vendor will also be more willing to improve the defenses when the newly discovered vulnerabilities are publicly known.

The hiding of the security design principles is not a good idea – this is called security through obscurity. In cryptography it must be assumed that eventually the design will become known to the opponent and it is much better if the design has been analyzed by many people before this happens.

Essential things:

The general workflow, how the keys are derived, standards used, used ciphers and their modes of operation.

## Obtaining a license

The program stops sending/receiving after the evaluation period of 60 days has passed unless a license is obtained. This happens also if a time based license expires. The stored emails can always be viewed.

Every computer that has EndCryptor's security database needs a license. One computer can have multiple security databases i.e. instances of EndCryptor (for different roles of the computer's user) – only one license is required for one computer. The license must be installed into every EndCryptor instance on the computer. A computer containing only the program code does not need a license. If a computer is a network server then a separate license is needed for every security database hosted on the server (we do not recommend placing the database on a server). A USB stick containing a security database needs a license, the stick may contain multiple security databases and only one license is required.

Licenses can be ordered using the program's order form or from product's website [www.endcryptor.com](http://www.endcryptor.com). If the order form in Menu: Tools->Licensing->Order is used the payment method is bank transfer. Use this form to order more than say 50 licenses. Licenses can be also purchased from the product website. Payment method is then credit card. Licenses purchased from the product website are delivered immediately upon payment.

If the order form in the program is used the order is encrypted and sent to [orders@endcryptor.com](mailto:orders@endcryptor.com). The order processing may take one working day. One person in the organization places the order and receives the license file which can contain many licenses. The license file is given to those in the organization who need it. To use 1 license from the license file select from Menu: Tools->Licensing->Receive license file.

There are two kinds of licenses: time based and version based.

Time based one or two year licensing gives the right to use the latest version and its updates up to the expiration date of the license. Renew existing time based licenses when there is less than 1 month to expiration. When the new license file is received the new licensed time is added to the end of the existing license in each such case. In other cases the license's starting time is its issue date.

Version based licensing gives the right to use a certain version and all its updates any number of years, a version based license to version 2.x is a license for all values of x.

It is possible to change the licensing scheme from annual to version based. One year's annual unit price is subtracted from each license's version based unit price in this case. All the installations which will use such a license must have used a time based scheme before.

Payment info if the payment method is bank transfer:

Enternet Oy

Finland

VAT number: FI 08210504

BANK: Nordea Bank Finland Plc, Helsinki

SWIFT: NDEAFIHH

IBAN Account number: FI08 1220 3000 2499 00

Technical details:

Individual encrypted messages do not contain user's values Enternet Oy knows. This implies that if Enternet Oy is shown an encrypted message created by a licensed customer then Enternet Oy cannot determine the customer in question.