EndCryptor

Version 2.5.5 www.endcryptor.com. Product of Enternet Oy, Finland.
TIP: Use Bookmarks for navigation

Contents

Date of this document: November 16, 2023

Overall description

Superior protection for the real world

EndCryptor protects old encrypted emails even if a hacker gets current encryption keys. EndCryptor is designed to protect old encrypted traffic and also to recover from attack – the attacker will lose the ability to decrypt new incoming emails.

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 before the year 2030 there will be computers that can break current classical public keys. EndCryptor uses classical public keys and new quantum attack resistant public keys. Note that otherwise current encrypted traffic can be copied and decrypted by quantum computers when they become reality.

State of the art cipher and public keys

The implementation of symmetric encryption and public keys uses publicly available source code developed by the scientists who designed the systems.

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. An 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 settings into EndCryptor.
- 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.
- Emails can be exported in eml format. They can be imported into email archiving solutions. They can also be viewed by many email client programs or dragged and dropped into an existing local email folder. 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 company-wide (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.
- Compression of plaintext. Required amount of random bytes are added to hide the length of this compressed plaintext - encrypted messages have different sizes even if their decrypted content is the same.
- A message may have more than one receiver. Contacts can be grouped.
- 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 will do the sending of the message.
- 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.

Quick start guide

If you want to send encrypted email to someone you must have that person’s Public Key Packet. Once you have it on your disk do drag and drop it into EndCryptor. You can also click the ‘New Contact’ button to receive the packet. The intended receiver must also have your public key packet inserted into EndCryptor.

To save your public key packet use Menu’s Tools→My Public Key Packet and save your packet.

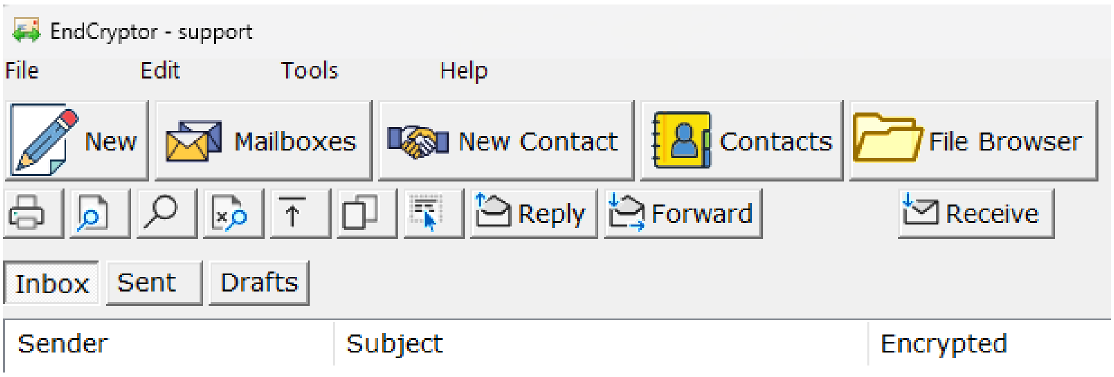
Note that if there is some group of people who need to use EndCryptor their public key packets can be zipped into one archive and this can be distributed securely (via protected website/file transfer) to members of the group. When a member receives the zip file it is dropped into EndCryptor and new contacts are automatically inserted.

When two persons start communicating with EndCryptor they both send their own public key packets to the other person. This can be done e.g. as an attachment in normal email. After this they can exchange encrypted emails. Note that the sender must have receiver's packet and the receiver must have sender's packet. The packet contains e.g. signature verification public keys by which a receiver can verify that an encrypted email is from the claimed sender.

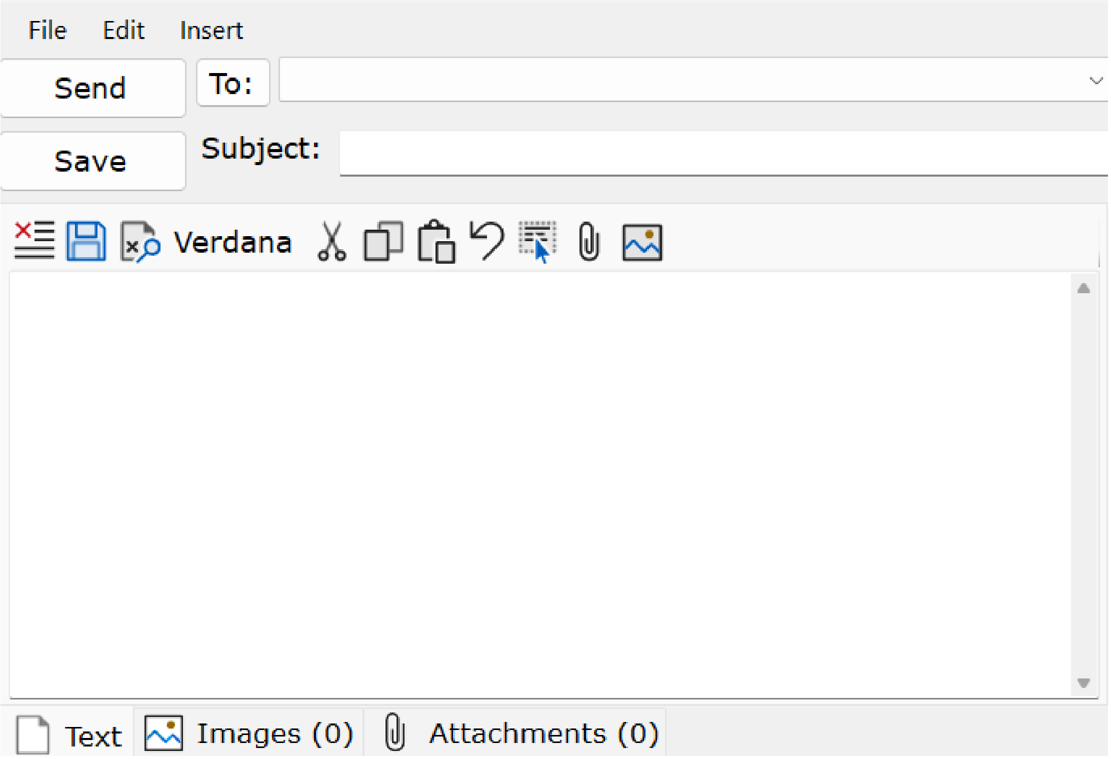
Now when at the beginning the public key packets are exchanged this must be done only with reliable other party. After all senstive information is going to be exchanged in encrypted messages. One must be sure that the other party is whom he/she claims to be and e.g. that the email address is the same as before and clearly belongs to the person and the organization in question. When a contact has been created its email address cannot be changed until some encrypted emails have been exchanged.

If it is considered possbile that there is an adversary who modified the public key packets during their transmittal then this be detected via Contacts->Advanced->Test for MITM after some encrypted emails have been exchanged.

Main window:



Example of email composing window:



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

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.

Post Quantum public keys use a construction called **K**ey **E**ncapsulation **M**echanism (KEM). In it a person A uses a public key of person B and encapsulates (encrypts) a shared secret into ciphertext. This ciphertext is sent to the person B who decapsulates (decrypts) it using the private key of the public key in question. The decapsulation produces the same shared secret.

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 the exposure of a private key? EndCryptor 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 new public keys every 2 weeks. These keys are contact specific - each contact will receive different new public keys when they exchange emails. When a person whose private key has been stolen sends a new message which has the new public keys 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.

Man-in-the-middle attack

This attack can happen when a person sends a public key packet to another person at the beginning of the communication. In cryptography this attack is traditionally explained using three persons: Alice, Bob and Mallory. Suppose now that Alice and Bob want to communicate securely and that there is a third person Mallory who wants to decrypt the exchanged messages.

Alice sends her 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 ← → Mallory ← → 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. Mallory decrypts and re-encrypts every email exchanged between Alice and Bob.

Traditionally this attack is prevented either by using certificates or using some communication between Alice and Bob to check that the first messages were not altered (i.e. to compare checksums of exchanged messages). The certification solution is used in online communication to websites (https which means TLS/SSL) – it has some risks, see ‘the risks of SSL’ chapter of this document. Email encryption solutions use either certificates (S/MIME or application specific) or comparison of checksums.

In this release of EndCryptor users can compare checksums to detect the attack.

When Alice and Bob start communicating with EndCryptor they both send their own public key packets to the other person. This can be done e.g. as an attachment in normal email. After this they can exchange encrypted emails. Note that the sender must have receiver's packet and the receiver must have sender's packet. The packet contains e.g. signature verification public keys by which a receiver can verify that an encrypted email is from the claimed sender.

If Mallory can modify the public key packets during their transmittal then he can start the man-in-the-middle hadling and thus decrypt the messages. This can be detected via Contacts->Advanced->Test for MITM after some encrypted emails have been exchanged.

Now when at the beginning the public key packets are exchanged this must be done only with reliable other party. After all senstive information is going to be exchanged in encrypted messages. One must be sure that the other party is whom he/she claims to be and e.g. that the email address is the same as before and clearly belongs to the person and the organization in question. When a contact has been created its email address cannot be changed until some encrypted emails have been exchanged.

Note that if there is some group of people who need to use EndCryptor their public key packets can be zipped into one archive and this can be distributed securely (via protected website/file transfer) to members of the group. When a member receives the zip file it is dropped into EndCryptor and new contacts are automatically inserted.

The risks of SSL

This chapter is about the risks of relying on browser based encryption (TLS/SSL/https) - 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'). **This chapter should interest people who are being marketed browser based email encryption solutions which do not need any software on user's computer.**

On March 2017 WikiLeaks published leaks from the hacking arsenal of the CIA (USA's Central Intelligence Agency). In some of those documents there is advice to malware writers of CIA: **'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 equipment: **"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 browser connects to a HTTPS - SSL or TLS server, the server sends a certificate to the browser 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.

All computers doing SSL/TLS have a special store that keeps the public keys of CAs. These public keys are inside certificates that are called the root certificates. When an incoming certificate from a web server is checked its validity ultimately depends on this root certificate - there must be a valid cryptographic signature chain from the root certificate to the incoming certificate.

Now anybody who knows the private key of a root certificate in user's computer can impersonate, decrypt and modify the SSL/TLS stream from every website to this specific computer using a technique called Man-In-The-Middle (MITM) - it will be explained later. **So, by design of the certification infrastructure, every publicly accepted Certificate Authority can - if it turns evil - do that to every SSL/TLS capable computer on this planet.** It simply issues a certificate for a server and gives the used public key's private key to an attacker - who now can attack this one server in question. The evil turned CA could also give the private key of the root certificate or a CA certificate signed by the root certificate to the attacker - who can now attack every SSL/TLS server by issuing fake certificates as the need arises.

One can argue that if the data that should be protected is sensitive enough then it is too big risk that one of the about 600 CAs e.g. turns evil, is hacked, has a bribed employee, or is forced by some government to reveal a private key. The Certificate Transparency project started by Google tries to minimize this kind of problem - it is explained later.

Certificate Authority Trustwave admitted on February 4, 2012 that they had given one private customer a CA certificate signed by Trustwave's root certificate inside a special machine which used a given private key to generate certificates for any website. This was done to decipher and monitor all of company’s online SSL/TLS communication regardless whether the users' devices used were company provided or not – because the certificate was issued by a Certificate Authority no additional certificates were needed on users’ computers. See https://www.theregister.co.uk/2012/02/14/trustwave_analysis/ .

Now also every other root certificate generated by anybody placed somehow into the certificate store of a computer causes the nullification of protection to this specific computer if the relevant private key falls to wrong hands.

The attacker can place his certificate to the store e.g. by forcing the user to do this, by evil maid, by evil customs officer or by malware. Some antivirus and parental control programs intercept and decrypt SSL/TLS by placing their own root certificate to the store. There have been situations where the private key of this certificate is the same for every user of the program - this compromises them all, a user finds the private key from his own computer and then can compromise the others.

Researches published a study on the problems created non-public CAs (which they call 'hidden root certificates') on the root store in paper "Rusted Anchors: A National Client-Side View of Hidden Root CAs in the Web PKI Ecosystem. In CCS ’21, November 15–19, 2021, Virtual Event, Republic of Korea". They conclude that "Through cooperation with a leading browser vendor, we analyze certificate chains in web visits, together with their verification statuses, from volunteer users in 5 months. In total, over 1.17 million hidden root certificates are captured and they cause a profound impact from the angle of web clients and traffic. Further, we identify around 5 thousand organizations that hold hidden root certificates, including fake root CAs that impersonate large trusted ones."

The placing of a certificate to the root store 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. Also WiFi connection points and internet cafes can do the MITM attack if they have access to a proper private key.

On July 17, 2019 the government of Kazakhstan started enforcing a policy where web browser users are forced to install a specific root certificate on their computers. When a selected user tries to connect to a selected server the connection is not allowed by the internet service provider unless the government issued certificate is being used. According to a study by the lab Censored Planet from the University of Michigan (https://censoredplanet.org/kazakhstan) the decryption targets included e.g. Gmail, Google, Facebook, Messenger, mail.ru, translate.google.com, Instagram and Youtube.

It can be very dangerous to use unknown computer's web browser to connect to a web server - the computer may have a special root certificate installed which enables MITM attacks and the computer may also route the internet traffic to a MITM machine.

There is also an attack using the DNS system of internet: the attacker succeeds in changing the DNS records in some of internet's DNS servers. After that the attacker requests new certificates for targeted domains from a CA. It can now act as the proper server. The attacker can also redirect the targets' traffic to attacker's servers. This kind of attack was revealed on November 2018 and targeted 50 Middle Eastern companies and government agencies. Use the term 'DNSpionage' to search for more information, see also https://krebsonsecurity.com/2019/02/a-deep-dive-on-the-recent-widespread-dns-hijacking-attacks/ by Brian Krebs.

Interesting attack scenario is also when the private key of the server has been exposed. The private key can e.g. be hacked, leaked by evil employee or can be forcefully obtained by authorities. If the server has not been configured to use **Perfect Forward Secrecy (PFS)** the attacker recorded old SSL/TLS 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 certificate is needed on user's computer since server's private key is known.**

The US. government forced an email service Lavabit to hand over SSL private keys probably to gather evidence against Edward Snowden. See https://www.wired.com/2013/10/lavabit_unsealed

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 SSL sessions (without PFS) can be opened using the exposed private key. See 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."

On January 14, 2020 Microsoft informed that a vulnerability was found on certificate validation. The flaw was reported to Microsoft by NSA, see https://media.defense.gov/2020/Jan/14/2002234275/-1/-1/0/CSA-WINDOWS-10-CRYPT-LIB-20190114.PDF. According to NSA "examples where validation of trust may be impacted include: **HTTPS connections, Signed files and emails** and Signed executable code launched as user-mode processes. **NSA assesses the vulnerability to be severe and that sophisticated cyber actors will understand the underlying flaw very quickly and, if exploited, would render the previously mentioned platforms as fundamentally vulnerable.**". This means an attacker using this flaw can e.g. start a man-in-the-middle attack without knowing any private keys. Cert coordination center (https://kb.cert.org/vuls/id/849224/) says "By exploiting this vulnerability, an attacker may be able to spoof a valid X.509 certificate chain on a vulnerable Windows system. This may allow various actions including, but not limited to, interception and modification of TLS-encrypted communications or spoofing an Authenticode signature.". Among affected versions is Windows 10 which was released on July 29, 2015.

Do you still remember what the leaked CIA papers warned: '**DO NOT solely rely on SSL/TLS to secure data in transit. Numerous man-in-middle attack vectors ...**'

How the Man-In-The-Middle (MITM) works? The attack can be done by any computer between the web server and the user which knows a proper private key (either root certificate's or server's). When the user initiates the connection to the server the attacker acts like the proper server - it can do this because it knows the private key. To the server the attacker acts like the user.

MITM attack: User ← → Attacker ← → Server.

Many companies use special machines to decrypt the SSL/TLS stream flowing in and out of the company. A certificate is placed on users' computers top enable this. This is done to e.g. search for malware from the stream. Citizen Lab’s report "Planet Blue Coat: Mapping Global Censorship and Surveillance Tools, January 2013" (from https://citizenlab.ca/publications/) 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.

The MITM attack can be tried on ‘normal’ SSL or TLS based email and webmail solutions and on email encryption solutions that are browser based. There are also Virtual Private Network solutions that use the web browser. Web browser based systems usually use marketing argument that no software is needed on user’s computer because only a web browser is needed.

Sometimes email encryption is provided in a total browser based way where there is an additional PGP encryption done by javascript. This does not prevent the attacker from being able to decrypt the PGP message. The attacker modifies the javascript that the server sends so that it e.g. sends PGP's private key to the attacker when run by the user.

EndCryptor has to use TLS when contacting email server. **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 TLS 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 well known and called certificate pinning.

The **Certificate Transparency project** (https://www.certificate-transparency.org/) by Google tries to improve the certification infrastructure. This project tries to log all CA issued SSL/TLS certificates in the world in order to make it possible to detect hostile certificates issued to a specific server. Major CAs take part of it and also search engines may submit certificates they see into the Certificate Transparency logs. Certificates issued after April 30, 2018 will not be accepted as secure by the Chrome browser unless they have a signed statement (a Signed Certificate Timestamp (**SCT**) accompanying the certificate) "Specifically, Certificate Transparency makes it possible to detect SSL certificates that have been mistakenly issued by a certificate authority or maliciously acquired from an otherwise unimpeachable certificate authority. It also makes it possible to identify certificate authorities that have gone rogue and are maliciously issuing certificates. "

If a publicly accepted CA creates a certificate for a server then it must be logged into the transparency logs. 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 wrong certificate can be misused to decrypt/modify server's traffic until it is detected and revoked and the users know about the revocation. One can draw a conclusion that there must have been serious misuse cases because such a big system is needed.

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.”

The Certificate Transparency logging and protection of certificates is not done to local certificates that are not created by a publicly accepted CA 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. A browser that does

not accept a certificate without the required SCT from a public CA will accept a certificate without the SCT if the certificate is not from public CA.

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.

Some examples of interesting scenarios:

A 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', Financial Cryptography and Data Security '11 March 2011. They define an attack where a government forces a CA to issue a certificate to a government given public key in order to decrypt traffic of a selected server.

Certificate Authorities can be targeted by viruses, e.g. Duqu virus targeted certificate authorities and used stolen and forged certificates for its purposes. Electronic Frontier Foundation's SSL Observatory project (https://www.eff.org/deeplinks/2011/10/how-secure-https-today) (2011-10-27) 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**.”

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. See (https://security.googleblog.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.

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 this was a violation of their procedures. See (https://security.googleblog.com/2013/12/further-improving-digital-certificate.html)

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.

Cryptographic technical details

Both parties that send and receive emails need that EndCryptor is installed on users' computers in order to encrypt and decrypt.

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 user's email system. 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.

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 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 are the Edwards curve Ed25519 and the Curve25519. The Edwards curve is used for signing and the Curve25519 for Diffie-Hellman calculation.

Encryption keys are determined using public key technology (classical: Curve25519, quantum attack resistant: FrodoKEM-1344 and mceliece6688128f (round 4)). Note that these variants of Frodokem and Mceliece are recommended by Germany's Federal Office for Information Security, see BSI TR-02102-1. The hybrid key setup is done according to the 'Concatenate hybrid key agreement scheme' of ETSI TS 103 744 V1.1.1 (ETSI is a European Standards Organization (ESO). .. the recognized regional standards body dealing with telecommunications, broadcasting and other electronic communications networks and services.)

Signatures are classical Edwards curve (Ed25519) signatures and CRYSTALS-Dilithium3 signatures. NIST of USA recommends CRYSTALS-Dilithium as the primary algorithm for signatures. Only the first messages use signatures, later ones are authenticated using Keccak (SHA3-256) macs that use keys derived from a shared secret. Signatures and macs are calculated over sha2-512 hash of encrypted data.

The implementation of the Ed25519, Curve25519, and Chacha20 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 FrodoKEM-1344 is from GitHup: PQCrypto-LWEKE. The code for mceliece6688128f is from the round-4 NISTPQC submission package. Dilithium code is from GitHup: pq-crystals/dilithium .

The private keys of public keys are made using BCryptGenRandom system call and private keys of classical elliptic keys are made by 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.

Protocol outline:

This is a stateful protocol. For each contact there is a state for sending and a state for receiving. When encrypting and decrypting the corresponding state is consulted. Keccak algorithm shake256 is used to form a new state and encryption/decryption keys for current message. Upon sending user periodically creates new contact-specific public keys, these public keys are inserted into the outgoing message. When sending a message, a received public key is used once to generate a shared secret (by creating a classical Curve25519 public key and post quantum KEM ciphertext), the outgoing message will have these public elements so that the receiver can calculate the same shared secrets.

In the following the Sender is going to send a message to the Receiver:

If the Sender has not received a message from the Receiver before, then the Sender must somehow get a public key packet from the intended Receiver. The packet contains Receiver's Curve25519 and Frodokem public keys as well as Receiver's Ed255119 and Dilithium3 keys.

If the Sender has not received KEM calculated ciphertexts against his last sent public keys: If the Sender deduces that the Receiver may not have received his last sent public keys the Sender inserts them into the message.

If the Sender has received KEM calculated ciphertexts against his last sent public keys: If two weeks has passed since the last new Curve25519 and Frodokem public key generation then those keys are generated and inserted into the message. Every fourth package of new public keys also contains a new mceliece public key. A group send can contain only 1 mceliece public key (due to its 1MB size).

If the sender has available received public keys from the Receiver: Shared secrets are created and a new curve25519 and required KEM ciphertexts are created. Let State be sender's current state for sending and Shs the calculated hybrid shared secret. Let X be a 96 bytes shake256 result calculated over shared secret Shs and State. The first 32 bytes of X are the new state for sending and next 32 byte blocks are the encryption and mac keys for the message to send. The hybrid key setup is done according to the 'Concatenate hybrid key agreement scheme' of ETSI TS 103 744 V1.1.1. The used received public keys are deleted.

If the Sender has no public keys available from the Receiver: Let X be a 96 bytes shake256 result calculated over the State. The first 32 bytes of X are the new state for sending and next 32 byte blocks are the encryption and mac keys for the message to send.

The message is signed if no messages have been received from the message's Receiver otherwise there are only macs.

When receiving a message there are two macs that are calculated. The one that is checked first uses a previously calculated key stored into Receivers database, the second mac checking uses the key that is the outcome of a state's processing. Let Y be a 32 byte shared secret obtained from hybrid key shared secret calculation (different than the Shs above) during send. The Sender stores this to be used as the first mac key when calculating the first mac to check incoming message. The parties communicating can deduce the proper key to fetch from the database based on its numbering. Note that this Y value is calculated only when there is a shared secret available. The second mac key is calculated during every encryption/decryption calculation and is the last 32 byte block of X.

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. The Personal Export key is a symmetric Chacha20 256 bit key. Additionally a Company Export Key can be used. Such public key is company issued and user installed mceliece6688128f public 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 by personal Export Key from backup media. If the company public key is used to encrypt backups then the KEM calculated ciphertext (208 bytes) is stored into the backed up file together with field F' which contains the actual encryption key encrypted with the key that is protected by the KEM ciphertext.

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 verity 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. 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.

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 congaing 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. 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. 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.

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.