

EndCryptor Crypto Station

Version 2.2.5 www.endcryptor.com. Product of Enternet Inc., Finland.

Contents

Overall description	2
Security features explained	3
EndCryptor's method of renewal of public keys	6
Detailed description of properties when the security database has been exposed	7
Cryptographic technical details	9
Tutorial on public keys	10
To: Really security conscious user	12

Overall description

Protects messages sent over the net. These messages are typed using EndCryptor and they may contain files (attachments).

Both the sender and the receiver must have EndCryptor installed. Also they need an email or another program that actually delivers the message from machine to machine. If this program is Microsoft Outlook then the sending and receiving is maybe most convenient to the user – in other cases the user must perform more actions when receiving messages. The encrypted message is a file that can be sent/received by any program that can send/receive files.

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, moved between different user creatable mailboxes and exported in cleartext form. The export feature allows the user to have a complete cleartext archive of the communication.

EndCryptor puts special care to the security of a message during its traversal in the net. The aim is to protect the message even if a hacker or a spy program gains access to a user's computer and tries to use thus obtained information to decrypt at some point in time captured earlier messages. Also EndCryptor is designed to recover from this kind of attack – the attacker will lose the ability to decrypt new incoming messages. These new features in email encryption software market are explained in more detail later in this document.

Encryption is done using the AES cipher, with key length of 256-bits. Public key technology is based on ANSI standard. The protocol that provides the message protecting features during the traversal in the net has been published under the Patent Cooperation Treaty.

No public key infrastructure is required although public keys are used.

Requirements:

Windows XP (SP2 or later), Vista or Windows 7.

If Microsoft Outlook is used then its version must be Outlook 2003 or later.

If another email client than Outlook is used then Simple MAPI support like e.g. in Mozilla Thunderbird, Outlook Express or Windows Mail eases the usage of EndCryptor.

Security features explained

EndCryptor offers features that are new on the email encryption software market: **backward security** and **recovery from an attack**.

Backward security means that if a hacker or spy program steals the current security data (encryption keys) he cannot use this information to decrypt earlier encrypted messages. The **recovery from attack** means that after some time the intruder has no use of the obtained security data if he tries to decrypt new messages created after the intrusion. This restoration of security is important since the victim of intrusion may be totally unaware of the adversary's activities. Even in case of a successful hacker's attack certain kind of **protection against identity hijacking** can be offered. EndCryptor uses public key encryption technology, however, **no public key infrastructure (pki) is required – there are no costs due to pki**.

Without backward security 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 cryptosystems also all communication sent **from** you is exposed! 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 current computer environment where worms, viruses and spyware try to access every computer in the net it is essential to have an encryption solution that is prepared to face attacks.

<i>Comparison between EndCryptor and the PGP-family of email encryption products (PGP, OpenPGP, GnuPGP,...) in case of a successful spying attack</i>		
	EndCryptor	PGP -family
Backward security (= are encrypted messages sent to the victim before the attack protected?)	YES	NO
Recovery from the attack will happen	When the next message from the victim is decrypted.	When the certified new public key of the victim is received. This usually happens at predetermined intervals - after several months.
Identity hijacking (= identity theft) will be revealed	YES	NO

The spying attack^{1,2} 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 (or whatever it is called) and the password's and the database's transmittal to the attacker. This exposure of the security database can happen other ways also: the user turns **from friend to foe** and reveals the 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.

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 a 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 aim is to minimize the amount of then exposed data.

Detailed Features

- Both the sender and the receiver must have EndCryptor installed. Also they need an email or another program that actually delivers the message from machine to machine. If this program is Microsoft Outlook then the sending and receiving is handled most conveniently by EndCryptor and Outlook – otherwise the user must perform more actions upon receiving a message. The encrypted message is a file that can be sent/received by any program that can send/receive files. If another email client than Outlook is used then Simple MAPI support like e.g. in Mozilla Thunderbird eases the usage of EndCryptor.
- Encryption is done using 256-bit keysize AES (**A**dvanced **E**ncryption **S**tandard of **U**SA).
- Encryption keys of messages are determined using **elliptic curve public key technology**. According to the National Security Agency of USA "The best assured group of new public key techniques is built on the arithmetic of elliptic curves. ...as one scales security upwards over time to meet the evolving threat posed by eavesdroppers and hackers with access to greater computing resources, elliptic curves begin to offer dramatic savings over the old, first generation techniques." (In: The Case for Elliptic Curve Cryptography: www.nsa.gov/business/programs/elliptic_curve.shtml)
- Each message is digitally signed. This ensures to the receiver that the message was created by the claimed sender and that the file was not altered during traversal.
- 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

¹ To see the techniques attacker uses search the Web for: keylogger, spyware, fedware, "**Internet surveillance through a wiretap**", computer espionage, hacking.

² To see an attack designed against a specific encryption package, search the web for the **caligula virus**, this attack did not use a keylogger, but was a proof of concept attack.

EndCryptor has been given. As an option the user can decide if the incoming attachments in messages are stored also in plaintext form or only in cryptotext form. The stored messages can be searched, moved between different user creatable mailboxes and exported in cleartext form. The export feature allows the user to have a complete cleartext archive of the communication.

- **Properties when the security database of user Alice is exposed:**
 - 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 hijacking: either the hijack attempt fails or it succeeds but then all future messages exchanged between the fooled party and Alice will be rejected. Protection against id 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 replay attack where an adversary captures 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.
- After the decryption the correctness of the plaintext is verified using message authentication code.
- 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	1,226,337	2.11
bible.txt	4,047,392	855,829	1.69
world192.txt	2,473,400	476,516	1.54
kennedy.xls	1,029,744	132,378	1.03

bpc = bits per character (byte). EndCryptor was used with the default setting: at most 1000 random bytes were added to compressed plaintext.

- A message may have more than one receiver. Contacts can be grouped.
- When a new contact is added both parties send to each other one special initialization file and then they may verify using a cryptographic checksum that they received the files unaltered. These files contain public keys that initialize the communication between these two parties. Later when the parties communicate every message EndCryptor encrypts contains new public keys of the sender created at the time of sending. This method means that there is **no requirement to have a public key infrastructure (pki) for key renewal and revoking**. Our solution is more

secure than a pki based solution if the private keys are stolen. **EndCryptor renews the public keys at no cost which may not be the case if a pki based solution is used.**

- 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 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.
- 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 removable media must provide at least 520 MB of storage.

EndCryptor's method of renewal of public keys

A public key infrastructure (pki) means that there is a Certificate Authority that digitally signs every new public key. Users have the public key of the Certificate Authority and use this public key to verify the new public keys. When new public keys are introduced they must first be certified by the Certificate Authority.

EndCryptor delivers the new public keys of the sender in every encrypted message. When the message has been decrypted the new public keys are ready to be used. The encrypted message that delivers the new public keys is signed by a previous public key that the receiver is known to have. The public keys are specific to the receiver: if say Alice has many contacts then each one of them uses a different public key of Alice when a message is sent to Alice.

The first public keys are delivered in a special initialization file when a new contact is added. If Alice adds Bob as a new contact then Alice sends to Bob one initialization file. Bob receives the file and sends another initialization file as a reply back to Alice.

After the initialization new public keys are exchanged in encrypted messages whenever Alice and Bob communicate.

Then at some point in time Alice and Bob may contact each other and verify that they really are Alice and Bob (in the Internet it is possible to fake the sender of email) and that they received the files unaltered. A checksum is displayed and that number must be the same for the both parties. This verification can be done e.g. using telephone conversation. When a contact has been verified its state is **verified** otherwise the state is **non-verified**. When sending to a contact the state of the contact is shown.

Using our method the public keys are changed more frequently: if the parties communicate in turns one public key is used only once.

Our solution for the renewal of public keys is cost-free.

Detailed description of properties when the security database has been exposed

EndCryptor has a security database where it stores the public keys received and the private keys created. This database is encrypted and accessible if the user's entry password to the database is known. We now explain in detail the situation that arises e.g. **if a hacker Eve succeeds in accessing the computer of say Alice** who communicates with Bob. The attacker now succeeds in installing spying software (e.g. key logger) and then tries to utilize the security database, which she has transferred to herself. Note that this kind of attacks have in real life been demonstrated against encryption software packages - a known example being the Caligula virus. This exposure of the security database can happen other ways also: the user turns **from friend to foe** and reveals the data to the adversary; or is **forced** (e.g. by a court order) or **lured** to reveal current security data; etc.

Decryption of messages

Eve's possibilities to decrypt before (old) or after (new) this exposure transmitted and captured messages are limited:

- **Old and new encrypted messages sent from Alice can't be decrypted by Eve.**
- Old encrypted messages sent from Bob to Alice that have been decrypted by Alice cannot be decrypted by Eve - this is called **backward security**.
- New messages sent from Bob to Alice can be decrypted by Eve until Alice sends a message to Bob and Bob decrypts this message. Encrypted messages now sent from Bob to Alice cannot any more be decrypted by Eve - **EndCryptor has recovered**. If exposure of the security database is suspected, using this feature it is possible to achieve a secure state by sending a dummy message to the sender of an important message before the important message is encrypted and transferred - the time window for successful attack will thus become very narrow.

Note that if messages are exchanged in turns then the number of messages from Bob that the attacker can decrypt is either zero (Alice sent a message to Bob just after the intrusion) or one (Alice sent a message to Bob just before the intrusion) until the attacker has to perform a new hacking intrusion.

Identity hijacking

Because Eve has the security database and its password she may try to send messages to Bob and pretend herself being Alice (Eve uses a special program available in the Internet that fakes the sender of email).

- If Alice sends a message to Bob before Eve does then all Eve's messages are rejected by Bob's EndCryptor - **the hijacking attempt has failed**.
- If Eve is the one who sends the first message to Bob then it is accepted by Bob's EndCryptor but then Alice's all further messages will be rejected by Bob's EndCryptor. Also all Bob's messages to Alice are rejected by Alice's EndCryptor (but accepted by Eve's). **If messages are being rejected it should alarm the true holder of the identity.**

Protection against identity theft is important since a user may have blind reliance on the protection given by a digital signature, which assures that the message comes from Alice. As shown above an adversary may steal information and then try identity theft.

The above-mentioned features are achieved by cryptographic means. An outline of the used methods:

- **Backward security:** Every EndCryptor message is encrypted with different AES 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.
- **Recovery from attack:** Every message EndCryptor encrypts contains new 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.
- **Identity hijacking will be revealed even under spying attack:** the stored security data that is used to build a symmetric key changes after every decryption and depends on the just decrypted message.

In most encryption systems:

- An exposure of the victim's security data leads to the situation where all previous encryptions to (maybe also from) the victim can be decrypted
- The theft of the exposed security data opens the door for really fooling the victim's contacts.

Cryptographic technical details

Both parties that send and receive messages need that EndCryptor is installed in order to encrypt and decrypt. No third parties are used (e.g. to provide public keys, to provide online connection to a third party machine, etc.) neither an online Internet connection between the sender and the receiver is needed. 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 elliptic curves used are defined in American National Standard for Financial Services X9.63–2001 and are those created verifiably at random. The elliptic curve field size used in initializing a contact is 571 bits and the field size used in a public key in encrypted messages is 193 bits. According to current understanding the cryptographic strength of the 571 bit elliptic curve equals that of a 15'360 bit RSA size and that of a 256 bit symmetric block cipher key. Elliptic curve Diffie-Hellman algorithm is used to compute a shared secret, elliptic curve digital signature algorithm to sign and elliptic curve signature verification to verify a signature.

EndCryptor delivers all but the first public keys in encrypted form. They are encrypted together with the plaintext. An encrypted message contains one visible (not encrypted) public key created at encryption time, its Diffie-Hellman counterpart is identified using a number. An encrypted message is signed by a previously delivered public key that the receiver is known to have.

The 256-bit key size AES encryption is done in CBC mode (128-bit block size).

The cryptographic hash function is a so-called Davies-Meyer construction with Merkle-Damgård strengthening from a block cipher Rijndael (AES), key and block size is 192 bits. This kind of block cipher based construction is usually slower than a specific dedicated hash algorithm. On the other hand the construction's security is that of the used cipher's - which in the case of the industry standard AES is being carefully studied and monitored by the crypto community. We remind that the last dedicated hash algorithm to be cryptographically successfully attacked is SHA-1.

The plaintext ends with a CBC-MAC (Rijndael (AES) 256-bit key size and 192-bit block size), the MAC-key is different than the encryption key. During encryption both the plaintext and the MAC are encrypted. After the decryption the MAC is calculated over the plaintext and checked.

The private keys of public keys are made using a Goldreich-Levin hard-core bit generator. The initial seed consists of events like mouse movements and system's state.

Security professionals wishing to evaluate the protocol should consult the World Intellectual Property Organization (WIPO) web site for published Patent Cooperation Treaty applications, the early publication date is November 18, 2004, application number PCT/IB2004/051670 titled "ENDS - Messaging protocol that recovers and has backward security". This protocol is licensed from Pisaramedia Inc., Finland.

Tutorial on public keys

We use public keys for these reasons:

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

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. 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? We solve 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.

EndCryptor creates a lot of public keys. Each encrypted message contains new public keys of the sender. 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 solution those encrypted messages that the attacker has captured and the proper receiver decrypted prior to the attack cannot be decrypted by the attacker.

Please note that the attacker may have collected the exchanged messages and naturally after successful stealing of private keys tries to decrypt all of them. 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 by also calculating the cryptographic hash value and then using the corresponding public key the verification is done.

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?

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 and/or the software in participants' computers. He could e.g. 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* it should from time to time be reinstalled (or EndCryptor.exe copied) from protected media or be run entirely from protected media or its cryptographic hash value (checksum) should be checked. EndCryptor checks that the cryptographic hash values of its own program files are as defined. 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). As explained an attacker can modify the program code to bypass this check. The methods described at the beginning of this paragraph should be used to prevent this. 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.

To reveal a highly advanced *man in the middle attack* the verification of a contact can be done at any time. The checksum values at step 3 of the verification are different if an attacker is actively operating a so-called man in the middle attack where he decrypts and again encrypts a message in the middle of the communication. An attacker can launch this attack when the contact the initialization files are exchanged or if the attacker obtains the exposed security data of both parties' computers at the same time or he does the modification of one party's security data properly. If there is not a man in the middle attack the digital signatures ensure that the message was created by the claimed sender and that the message was not altered. During a man in the middle attack the signature verification public keys and the message are provided by the attacker and thus the signature verification only ensures that the message came unaltered from the attacker. If an attacker is performing the man in the middle attack and wants to stop it then he has to modify the security data on one participant's computer. Also if the attacker is operating the man in the middle attack but misses one message then its receiver cannot decrypt this message.

The most secure usage 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. END OF DOCUMENT