CryptoTE Help

Timo Bingmann

February 2009

Contents

1 Introduction

1.1 Summary

CryptoTE is a text editor with integrated strong cryptography. It is based on the popular Scintilla widget and automatically stores text data in secure encrypted container files. Compared to other "password keeper" programs, CryptoTE does not force any structure upon your data: it works with plain ASCII text and does not require you to fill in grids, key-value attributes, descriptions etc.

2 About Encryption

2.1 Weak Passwords

Much advice is be given by different people about choosing a good password. Following the advice is often difficult and it is commonly ignored. This help tutorial suggests a simple method to keep your sensitive login information secure.

First: do not think of pass*words*, rather think of pass**phrases**. No encryption program can keep your data safe if you set the password to a plain English word or, even worse, some word connected with your surroundings or identity. There are many lists of bad but nevertheless frequently used passwords on the Internet: try search for “top 100 passwords” for some amusement.

Confronted with having to choose a longer passphrase most people will want to write it down. Next bad thing to do is to keep the slip of paper directly at the computer. Do not do this!

To give an idea of a good password: regard that Serpent uses 256 keybits, that is 256 bits of random information. An average English text has 1.0 to 1.5 keybits per letter.

So this whole sentence corresponds to only about 51 keybits!

The keybits (that is entropy rate) can be increased by using names, special symbols and other non-natural language elements. A randomly chosen lower- or uppercase letter has 5.7 keybits.

2.2 Advice on Choosing a Password

My method to learn a new password is very simple: I use the random password generator built into CryptoTE and generate a string containing random letters. I never include z/y in the password because they are mixed on German vs. English keyboard layouts.

Depending on the purpose, I only use lowercase characters and choose an appropriate password length: the generator will show you the theoretic keybits of the password. Adapt the length to your needs.

My container password is about 25 upper and lowercase letters. That is 139.6 keybits. A lot better than a simple sentence. Something like this: DUWHmnBunfVQNUeCdQxpHHdIJ

You think you cannot learn 25 random letters? **Try it!** Your memory is way better than you think. Learn it by frequent repetition:

I use CryptoTE daily to fetch some passwords and it always requires you to enter the password. Through this repetition **you too** will quickly learn your random letters. For the starting time (a week or so) you can write the letters down on paper, but keep that paper slip safe! My favorite place: my wallet. After two weeks: burn it.

If you think 25 letters is way too much: try starting with ten, e.g. rZl2jXybem. That is already 57 keybits.

2.3 Measures of Security

CryptoTE can keep your text safe, but you must consider the suitability of encryption for your purpose. Against whom are you keeping your passwords safe?

- To keep them safe against your children, roommates, co-workers, wife or husband casually using your computer: OK.
- To be sure that if your notebook or USB stick is stolen that no sensitive data can be read: very good idea, also OK.
- Against phishing or keyloggers: no chance, they will see the entered password.
- Against police / law enforcement officials: not good! They'll jail you indefinitely.
- Against vicious, evil people: no chance! They will use a crowbar **on you** instead of your computer.

3 Features

3.1 Overview

- User-friendly Scintilla text editing widget, the same as used by Notepad++.
- Edits secure private container files (see section 3.2) holding multiple text or binary files.
- Highly-secure Serpent (256 keybits) encryption (see section 3.3) of sensitive data.
- Automatic compression (see section 3.4) using zlib or bzip2 to reduce container size.
- Multiple user passwords (see section 3.5) can be specified to access a container.
- Fast user-interface: Quick-Find and Quick-Goto bars like Firefox's find. I use the program myself daily.

- Auto-Close the container after a user-defined period of inactivity.
- Built-in password generator to insert new passwords in the text.
- Portable, self-sufficient executable files for Windows and Linux available, very useful for USB sticks.
- Sleek wxAui tabbed interface from the newest wxWidgets (see section 3.6) version.
- Also usable from the Unix/Linux command line (see section 3.7) on a text console e.g. via ssh.
- Modularized and well-tested container processing library.
- Translated into German (volunteers for more languages wanted).

3.2 Encrypted Containers

An encrypted container has the extension `.ect`. It can hold multiple text or binary subfiles. The contained files are encrypted using strong cryptography and are unreadable by other programs than CryptoTE.

Use multiple subfiles to structure your sensitive data like "`WebSitePasswords.txt`" and "`EMail-Accounts.txt`".

The container file format supports built-in encryption (see section 3.3) and built-in compression (see section 3.4). It also supports multiple key slots for different passwords.

3.3 Built-in Encryption

CryptoTE contains built-in strong encryption. It uses a custom version of the Botan cryptography library.

While designing CryptoTE I decided not to burden a user (that is you) with a long list of encryption ciphers to select one. Instead I selected one for you, the strongest currently freely available one: **Serpent**.

Why use Serpent? Serpent was among the AES finalists and supports 256 key bits block encryption, that was a minimum requirement. The winner of the AES contest was Rijndael, probably because it is faster by a few percent. Serpent is a bit slower but supposedly more secure. As for the purposes of CryptoTE: encrypting rather small amounts of text or binary data, speed was not an important criterion. Instead Serpent was chosen because if someone ever finds a way to break Rijndael/AES then Serpent will (hopefully) still be safe for a short time. Even though both are based on the same cryptographic mechanisms, more cryptanalysis (read: attempts to break) will be directed at Rijndael/AES.

However secure encryption does not end with selecting a cipher. Instead it starts there: the key material must be stored securely, the contained key hashes must be irreversible.

One mistake in design of the container format can render the encryption weak or even breakable. Be aware that it is rather easy to make such a mistake and I am sure many other "password keeper" programs contain such errors.

It is also *very easy* to design a container that has a backdoor, i.e. that can be decrypted without the password. CryptoTE **does not contain such a backdoor**, there is **no viable method** to retrieve data without the password. I am sure many other "password protectors" contain such backdoors.

Each subfile of the container is encrypted using Serpent/CBC with a different randomly generated key and IV (initialization vector). The keys and IVs are stored in a global file table, which in turn is encrypted with a master key using Serpent/CBC. This master key is not stored in plain text within the container file.

Instead a container supports multiple decryption keys: Multiple Key Slots (see section 3.5). Each key slot contains an encrypted copy of the master key required to read the file table. The decryption key and CBC-IV for the master key can only be determined from the password entered by the user. The password entered by the user is hashed using PBKDF2 with HMAC(SHA256) as hash function. Two different random salts are used to generate decryption key and CBC-IV from the entered password.

More information about the container file format is available in the CryptoTE source code in libenctain/format.html.

I ran an extensive cryptography speedtest before designing CryptoTE: see `http://idlebox.net` for details.

3.4 Built-in Compression

As a bonus CryptoTE also contains automatic compression of text files. Nothing has to be activated: by default all files are compressed using zlib using the deflate algorithm.

Compression can be deactivated in the SubFile Properties dialog.

CryptoTE also contains bzip2 as alternative compression method. It generally only compresses really large text files better, smaller text files are handled better by zlib.

3.5 Multiple Key Slots

A container can be decrypted with multiple different passwords. This way multiple users can keep their password secret. This is called KeySlots in CryptoTE: a new password can be added in the menu entry “Password List”.

3.6 wxWidgets

Why choose wxWidgets as toolkit? For many reasons:

- It is cross-platform: versions of CryptoTE exist for Windows and Linux. MacOSX support is also possible.
- It looks “native” on all platforms: the frames and widgets look like the user expects them to look like.
- It allows me to compile a self-sufficient single .exe file for Windows, very useful for USB sticks.
- It is released under a very liberal license.

3.7 Command Line Interface

For Unix users and other power-users the CryptoTE program has a command line interface. This is very useful if you cannot start the graphical user interface, e.g. if logged in via ssh or if you are limited to a text console.

The best way to use CryptoTE without GUI is to start the “shell”: `cryptote -s file.ect`.

It will query for the container password and if decryption works a simple shell is started. Start using the shell by entering “help” for a list of command.

Be warned that using the “edit” command requires CryptoTE to save the contents in a temporary file outside the container. This file can then be modified using any text editor. If you wish to implement a built-in console text editor in CryptoTE, contact me.