Crypto Metrics Tool
v0.3.5
User Manual

Katarzyna Mazur

July 14, 2014

CMTool

Contents

1 CMTool's Overview

1.1 Interface Guide

Main window of the *CMTool* in GUI mode looks like below (see Fig. 1.1):

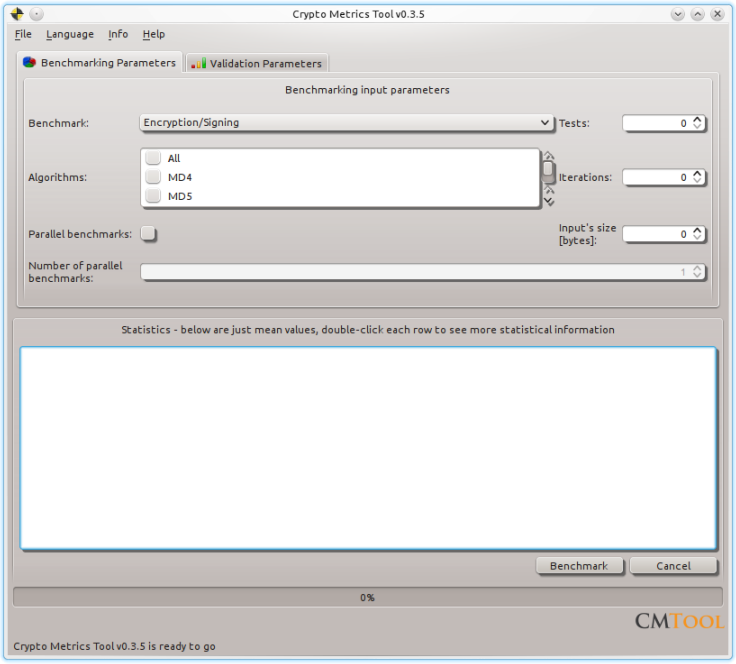


Figure 1.1: *CMTool's* main window

CMTool in GUI mode is written with the use of the open source version of *Qt* library - that's why the whole interface of the application should look almost exactly the same no matter if one uses it on Microsoft Windows or GNU/Linux. At the top of the main window one can see a menu with some options described in detail below:

1. Menu **File**:

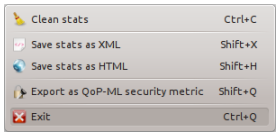


Figure 1.2: Menu **File** has 5 items

   (a) Item **Clean stats** - cleans information about calculated statistical primitives presented in a table located at the bottom of the main window
   (b) Item **Save stats as XML** - saves obtained results into the **.xml* file
   (c) Item **Save stats as HTML** - saves obtained results into **.html* files, makes simple documentation in html format
   (d) Item **Export as QoP-ML security metric** - cleans information about calculated statistical primitives presented in a table located at the bottom of the main window
   (e) Item **Exit** - saves obtained results into the **.qop* file, in format which can be understood by the *AQoPA*

2. Menu **Language** - lets user to change the language of the interface (not implemented yet)

3. Menu **Info** - lets user get some information about the hardware and software of her/his machine on which *CMTool* was run (see Fig. 1.3). Using keyboard shortcut, CTRL+I, one is able to see information about the software and hardware of the machine on a new tabbed window which immediately appears (see Fig. 1.4 and 1.5).

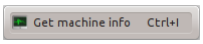


Figure 1.3: With *Get machine info* one can get information about her/his software and hardware

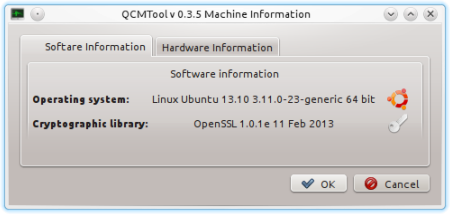


Figure 1.4: Information about the operating system and used cryptographic library.

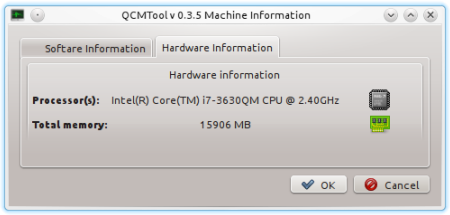


Figure 1.5: Information about the hardware of the machine on which *CMTool* was run.

4. Menu **Help**:

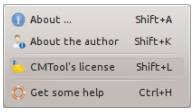


Figure 1.6: Information about the application.

(a) Item **About ...** - shows brief, general information about the *CMT* (SHIFT+A) (see Fig. 1.7)

(b) Item **About the author** - shows information about the author of the *CMT* (SHIFT+K) (see Fig. 1.8)

(c) Item **CMTool's license** - shows the CMTool's license text (SHIFT+L) (see Fig. 1.9)

(d) Item **Get some help** - shows this manual in a new window (CTRL+H)

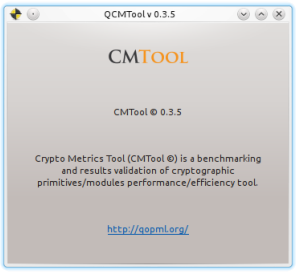


Figure 1.7: Information about the application.

Figure 1.8: Information about the author.

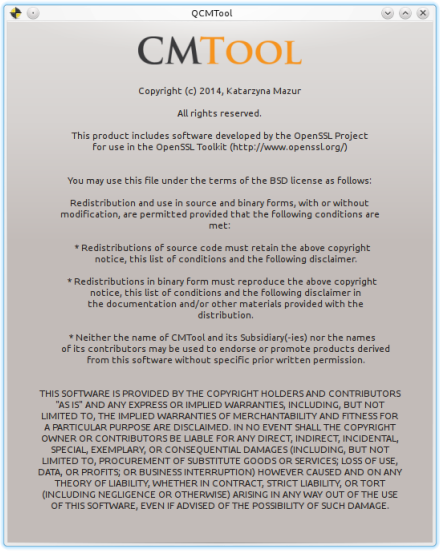


Figure 1.9: *CMTool's* license.

When user selects some cryptographic primitives for benchmarking (it can be done by checking appropriate boxex in a scrolled check box located on the main window (see Fig. 1.10), then she/he needs to define some input parameters. One can define how many test she/he wants to run, how many iterations should be done in each test, how big (in bytes) input message for benchmarking primitive needs to be (see Fig. 1.12). User is also able to choose which type of operation she/he wants to run - one can choose between encryption and signing (or decryption/verifying respectively) (see Fig. 1.10, 1.11).

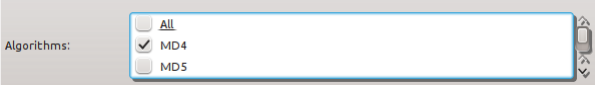


Figure 1.10: Choose some cryptographic primitives for benchmarking.

Figure 1.11: Choose operation type for benchmarking.

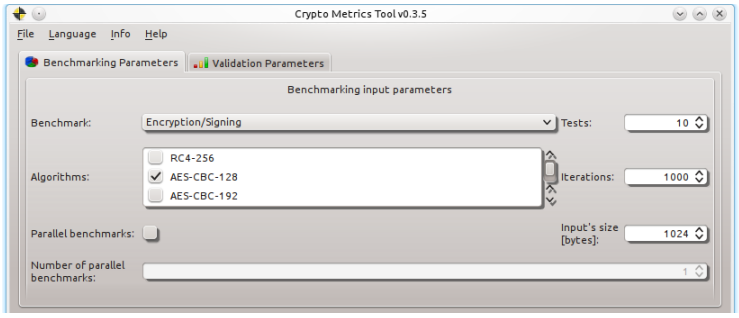


Figure 1.12: Available benchmarking parameters.

Besides choosing the benchmarking parameters, user is able to set some *validation* parameters. She/he can chose the *coefficient of variation (CV)* value to determine if the results gathered by the *CMT* are the time series, and run the test with the specific *confidence level*. The validation process is performed according to the steps defined in the *The robust measurement method for security metrics generation* article.

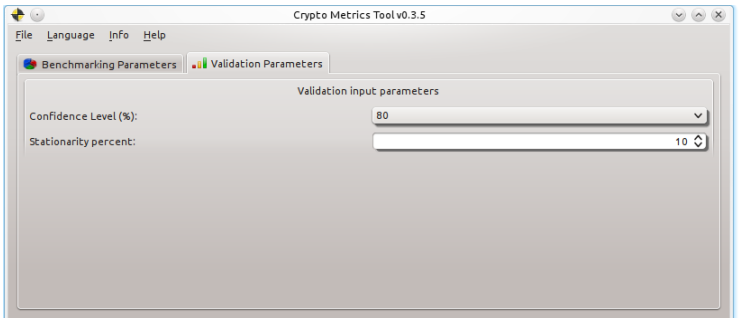


Figure 1.13: User can specify the *coefficient of variation (CV)* and the *confidence level* values.

Progress of performed tests can be seen at the progress bar located at the bottom of the *CMTool's* main window (see Fig. 1.14).

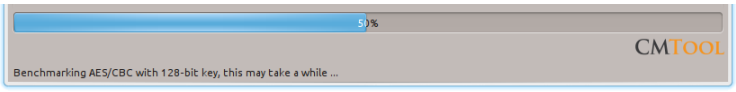


Figure 1.14: Benchmarking progress can be seen on the progress bar.

When all benchmarking is done, final results are presented. One can see mean values for every tested cryptographic primitive (base measure) summarized in a table at the bottom of the main window (see Fig. 1.15).

By double-clicking each row of the *CMTool's* table with collected results (mean values), one is able to see more statisitical information - mode, median, variance, standard deviation, kurtosis and skewness (see Fig. 1.16).

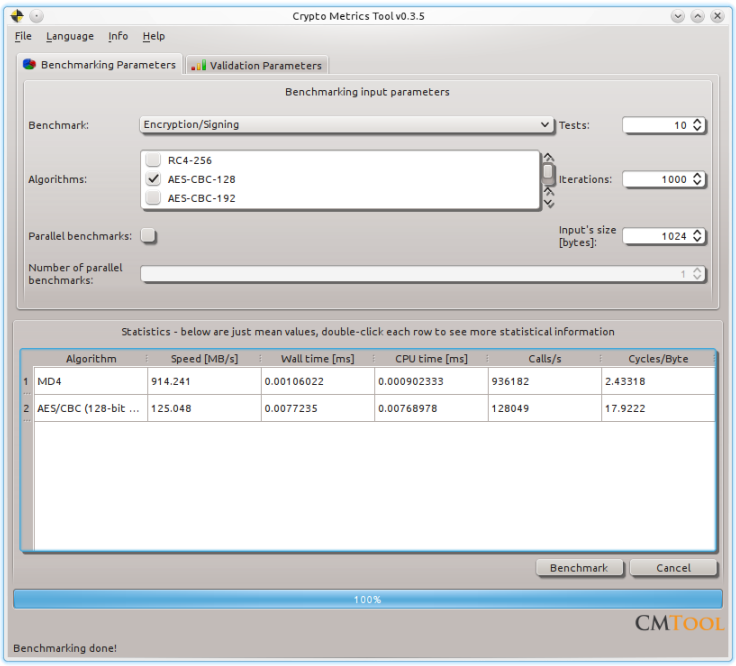


Figure 1.15: When the benchmarking is done, final results are presented.

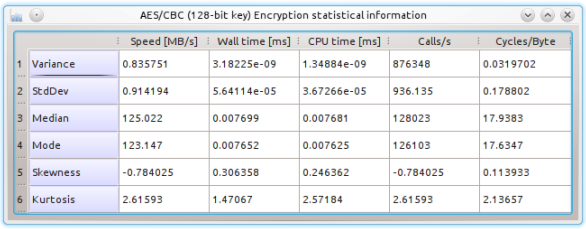


Figure 1.16: Measurement results are obtained, validated and presented to the user as statistical primitives.

2

CMTool's Installation Guide

2.1 Installation Guide for GNU/Linux-like Operating Systems

This chapter provides information about the installation of *CMTool* (simply, *CMT*) in console, as well as in GUI version on GNU/Linux-like operating systems.

2.1.1 CMTool Console Version - Installation

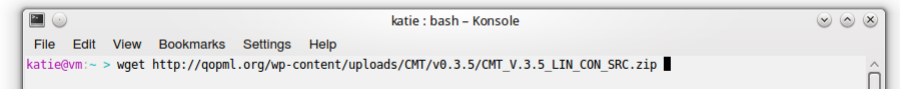
Example installation of *CMT's* console version was performed on Linux Mint 16 Petra and Fedora Linux 20 Heisenbug.

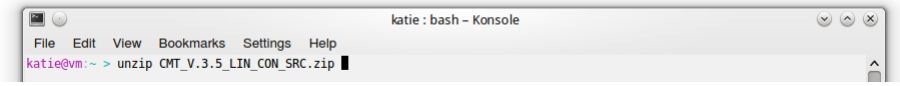
Installing *CMT* on Linux Mint 16 Petra

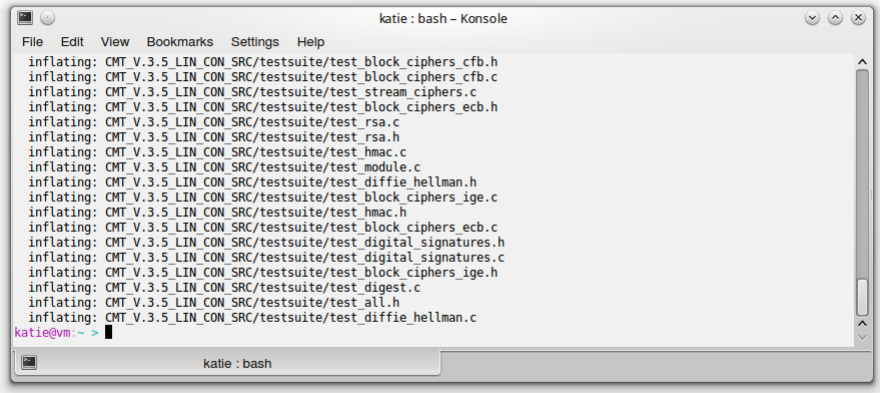
For GNU/Linux like operating systems, console version of *CMT* is shipped as a source package which you need to compile by yourself.

1. Download *CMT's* source package (`*.zip`) from QoP-ML's webpage:

```
cd ~
wget http://qopml.org/wp-content/uploads/CMT/v0.3.5/CMT_V.3.5_LIN_CON_SRC.zip
unzip CMT_V.3.5_LIN_CON_SRC.zip
```

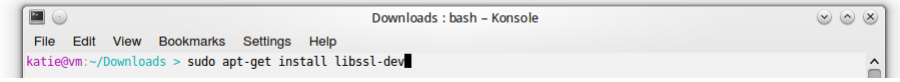






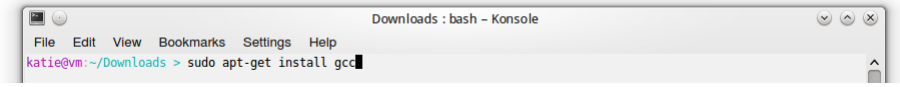
2. Since *CMT* uses *OpenSSL* for it's crypto benchmarks, to be able to compile and later use *CMT*, you should first install `libssl-dev` package. `Libssl-dev` is a part of the OpenSSL implementation of SSL, containing SSL development libraries, header files and documentation.

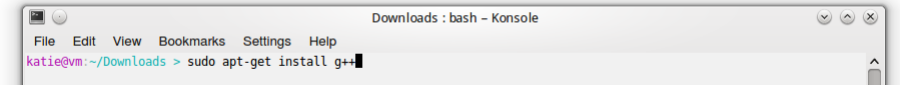
```
sudo apt-get install libssl-dev
```



3. *CMT*, as written in C/C++ languages, needs a C and C++ compilers to build executable files. Both `gcc` and `g++` will do the perfect job:

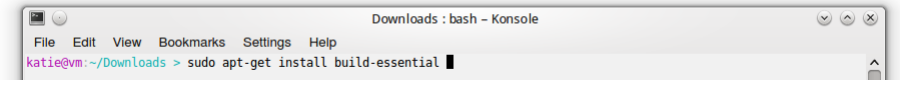
```
sudo apt-get install gcc
sudo apt-get install g++
```





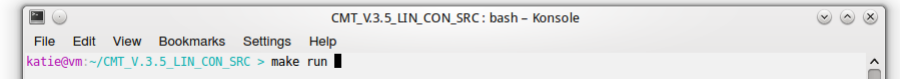
4. To perform compilation process in a a smooth and easy way, using single command from Makefile delivered along with *CMT*, install `build-essential` package. `Build-essential` package contains tools (like the gcc compiler, make tool, etc.) for compiling/building software from source.

```
sudo apt-get install build-essentials
```

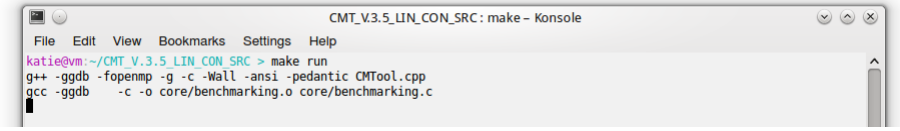


5. After installing all the required components to build *CMT*, you can actually compile it. Simply `cd` to the directory where you unpacked *CMT's* source (unpacking *CMT*, we assumed your `home/` directory):

6. Now simply type `make run` to compile and link *CMT*:



7. Below you can see the compilation process:



8. Compilation finished with no errors, meaning *CMT* has been successfully compiled and it is now ready to run (see Fig. 2.1).

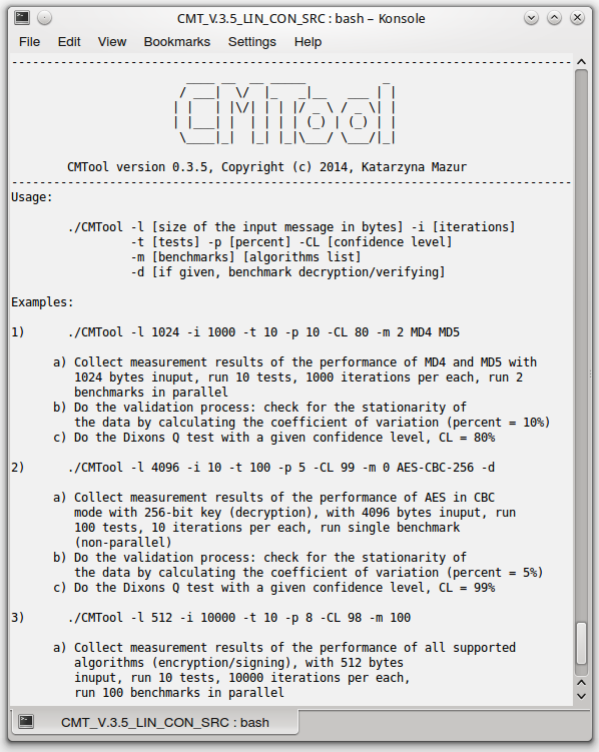


Figure 2.1: Do not be afraid about the `make: *** [run] Error 255` error, it is even not a real, actual error. It showed up because we ran (using the `make run` command) *CMT* without required parameters.

Installing *CMT* on Fedora 20 Heisenbug

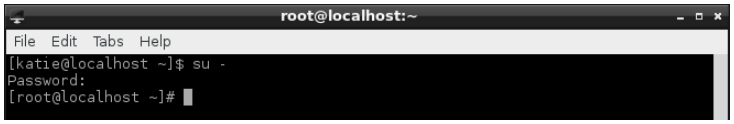
For GNU/Linux like operating systems, console version of *CMT* is shipped as a source package which you need to compile by yourself.

1. Download *CMT's* source package (`*.zip`) from QoP-ML's webpage:

```
cd ~
wget http://qopml.org/wp-content/uploads/CMT/v0.3.5/CMT_V.3.5_LIN_CON_SRC.zip
unzip CMT_V.3.5_LIN_CON_SRC.zip
```

2. Then, acquire `root` privileges:

```
su -
```



3. Since *CMT* uses *OpenSSL* for it's crypto benchmarks, to be able to compile and later use *CMT*, you should first install `Openssl-devel` package. `openssl-devel` is a part of the OpenSSL implementation of SSL, containing SSL development libraries, header files and documentation.

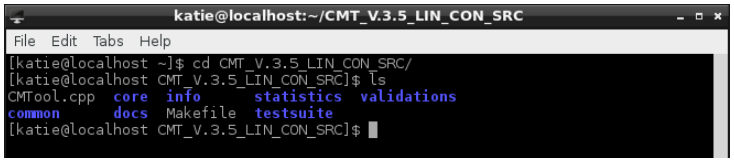
```
yum install openssl-devel
```

4. *CMT*, as written in C/C++ languages, needs a C and C++ compilers to build executable files. Both `gcc` and `g++` will do the perfect job:

```
yum install gcc
yum install gcc-c++
```

5. To perform compilation process in a a smooth and easy way, using single command from Makefile delivered along with *CMT*, install a group of `Development Tools` packages. `Development Tools` packages contain tools (like the gcc compiler, make tool, etc.) for compiling/building software from source. The `Development Tools` are a `yum group`, which is a predefined bundle of software that can be installed at once, instead of having to install each application separately. The `Development Tools` will allow you to build and compile software from source code.

6. After installing all the required components to build *CMT*, you can actually compile it. Simply `cd` to the directory where you unpacked *CMT's* source (unpacking *CMT*, we assumed your `home/` directory):



7. Now simply type `make run` to compile and link *CMT*:

8. Compilation finished with no errors, meaning *CMT* has been successfully compiled and it is now ready to run (see Fig. 2.2).

Figure 2.2: Do not be afraid about the `make: *** [run] Error 255` error, it is even not a real, actual error. It showed up because we ran (using the `make run` command) *CMT* without required parameters.

2.1.2 CMTool GUI Version - Installation

For GNU/Linux like operating systems, GUI version of *CMT* is shipped as a source package which you need to compile by yourself.

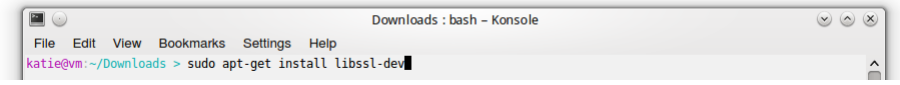
Installing *CMT* on Linux Mint 16 Petra

1. Simply download *CMT* from QoP-ML's webpage:

```
cd ~
wget http://qopml.org/wp-content/uploads/CMT/v0.3.5/CMT_V.3.5_LIN_GUI_SRC.zip
unzip CMT_V.3.5_LIN_GUI_SRC.zip
```

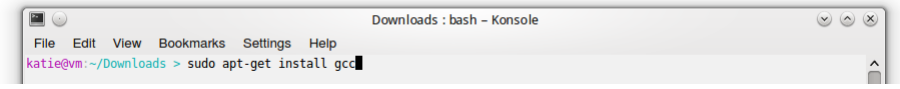
2. Since *CMT* uses *OpenSSL* for it's crypto benchmarks, to be able to compile and later use *CMT*, you should first install `libssl-dev` package. `Libssl-dev` is a part of the OpenSSL implementation of SSL, containing SSL development libraries, header files and documentation.

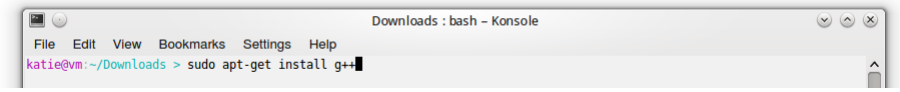
```
sudo apt-get install libssl-dev
```



3. *CMT*, as written in C/C++ languages, needs a C and C++ compilers to build executable files. Both `gcc` and `g++` will do the perfect job:

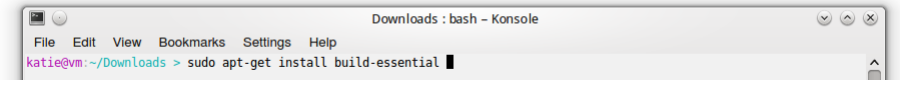
```
sudo apt-get install gcc
sudo apt-get install g++
```



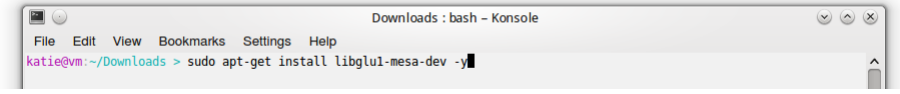


4. To perform compilation process in a a smooth and easy way, using single command from Makefile delivered along with *CMT*, install `build-essential` package. `Build-essential` package contains tools (like the gcc compiler, make tool, etc.) for compiling/building software from source.

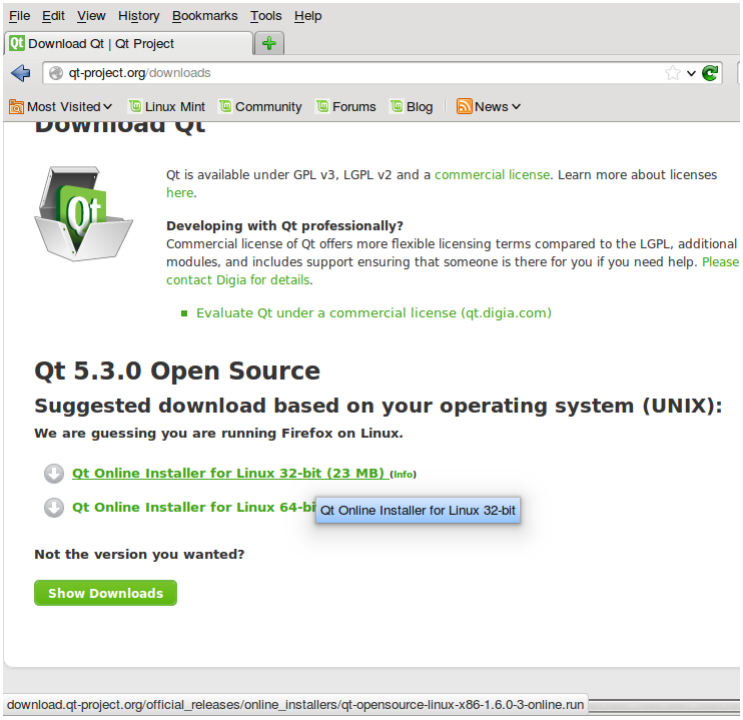
```
sudo apt-get install build-essentials
```

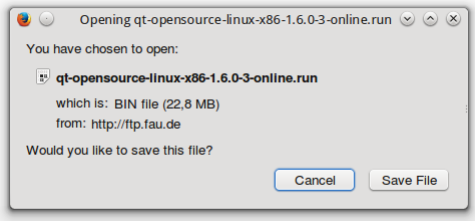


5. Install `libglu1-mesa-dev` package. The `libglu1-mesa-dev` package includes headers and static libraries for compiling programs with `GLU`. For a complete description of `GLU`, please look at the `libglu1-mesa package`.



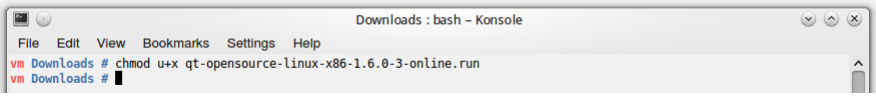
6. Download `Qt` libraries. The easiest and the fastest way of doing this is to download `QtCreator`:



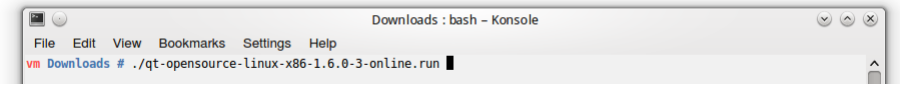


7. Install `Qt`. Make the installer file executable by typing:

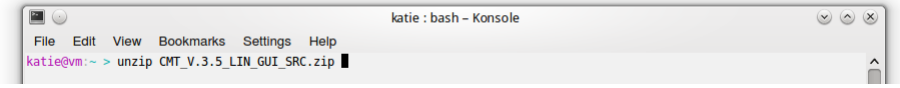
```
chmod u+x qt-opensource-linux-x86-1.6.0-3-online.run
```



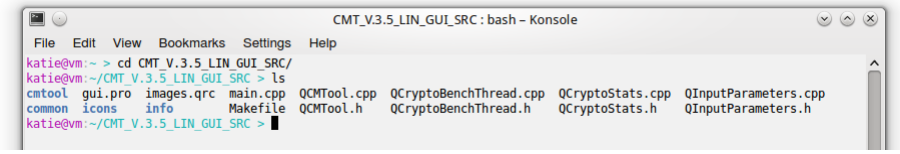
8. Run the installer as the `root` user and simply install `QtCreator`:



9. Extract *CMT's* archive:



10. Go to the directory where you extracted *CMT*:



11. Type `make` to compile *CMT* from source:

12. Type `./QCMTool` to actually run `CMT` in gui mode:

13. Alternatively, if you have any problems, install:

```
sudo apt-get install qt5-default
```

and then try to compile *CMT* (by typing `make` in the directory where you extracted *CMT*).

This package (`qt5-default`) sets `Qt 5` to be the default `Qt` version to be used when using development binaries like `qmake`. It provides a default configuration for `qtchooser`, but does not prevent alternative `Qt` installations from being used.

2.2 Installation Guide for Microsoft Windows-like Operating Systems

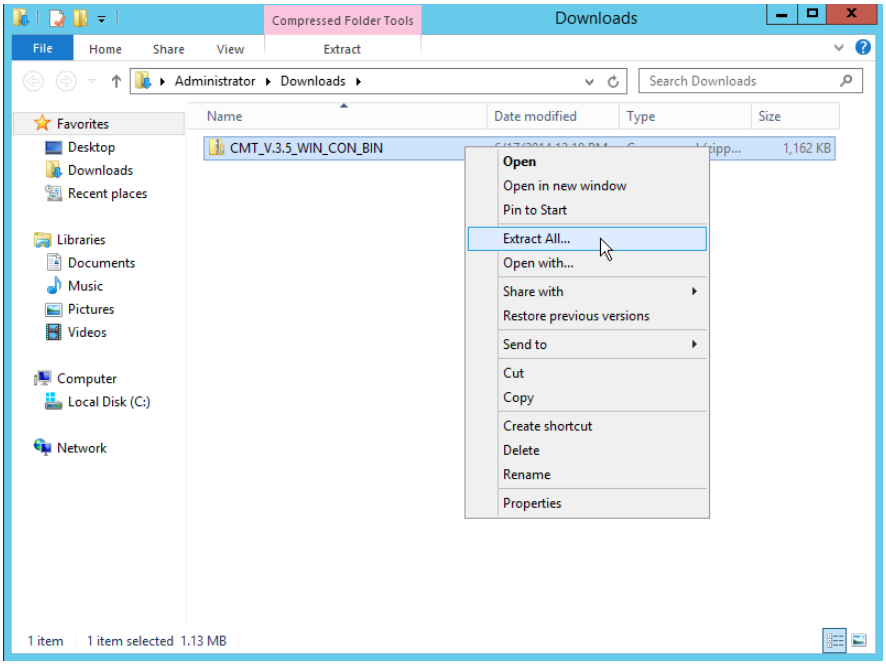
This chapter provides information about the installation of *CMTool* (simply, *CMT*) in console, as well as in GUI version on Microsoft Windows-like operating systems. Installation was performed on Microsoft Windows Server 2012.

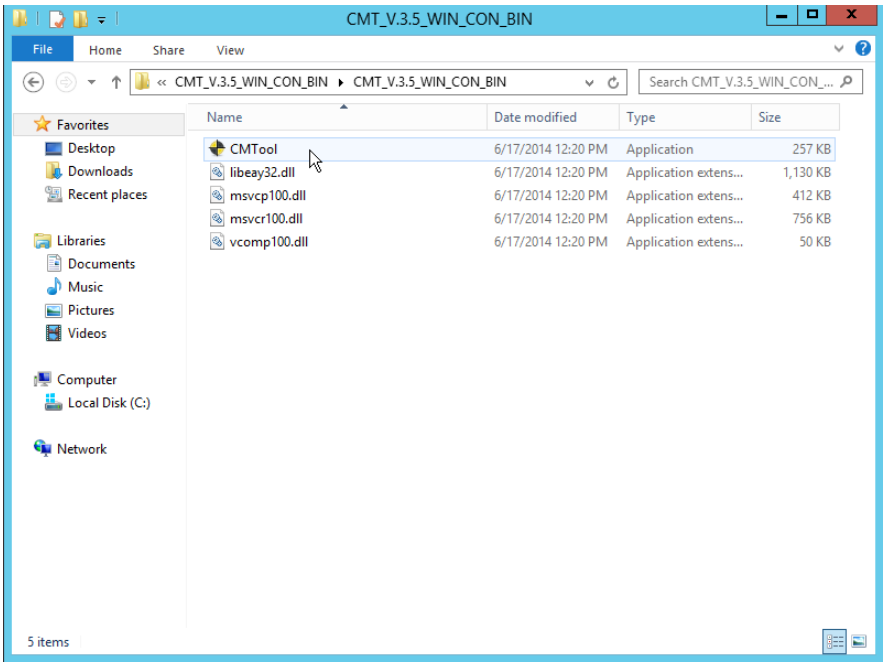
2.2.1 CMTool Console Version - Installation

Installing *CMT* on Windows Server 2012

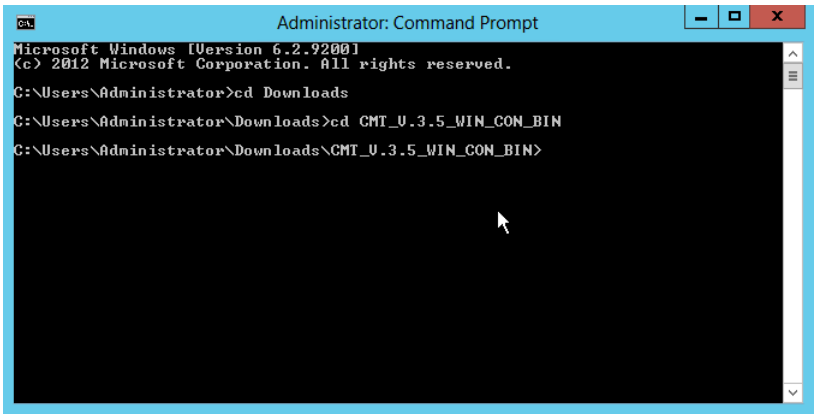
For Microsoft Windows-like operating systems, *CMT* is shipped as an executable application, so you do not actually need to install it.

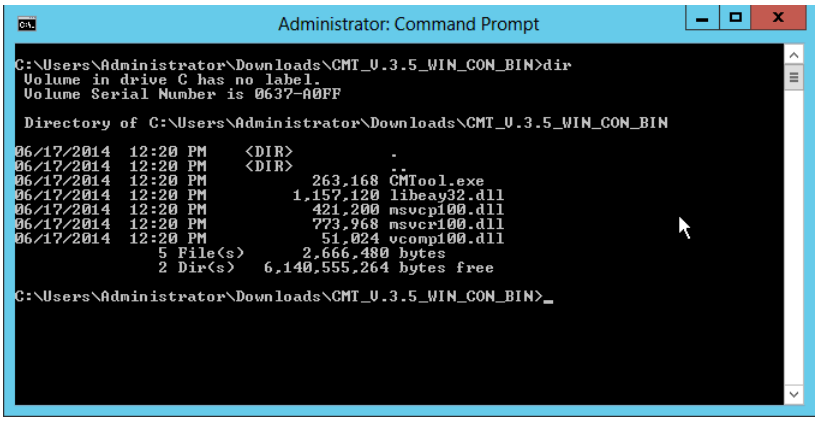
1. Simply download *CMT* from QoP-ML's webpage.
2. Extract *CMT's* archive:



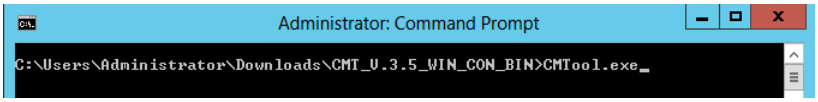


Open up `cmd.exe` and `cd` to the location, where you extracted *CMTool*:





3. Run *CMT* by typing `CMTool.exe`:

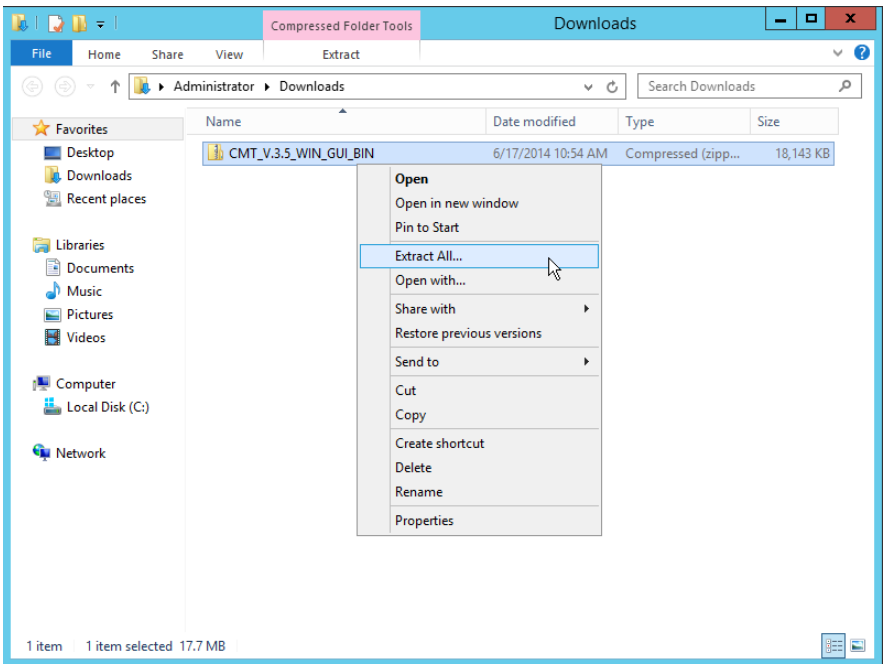


2.2.2 CMTool GUI Version - Installation

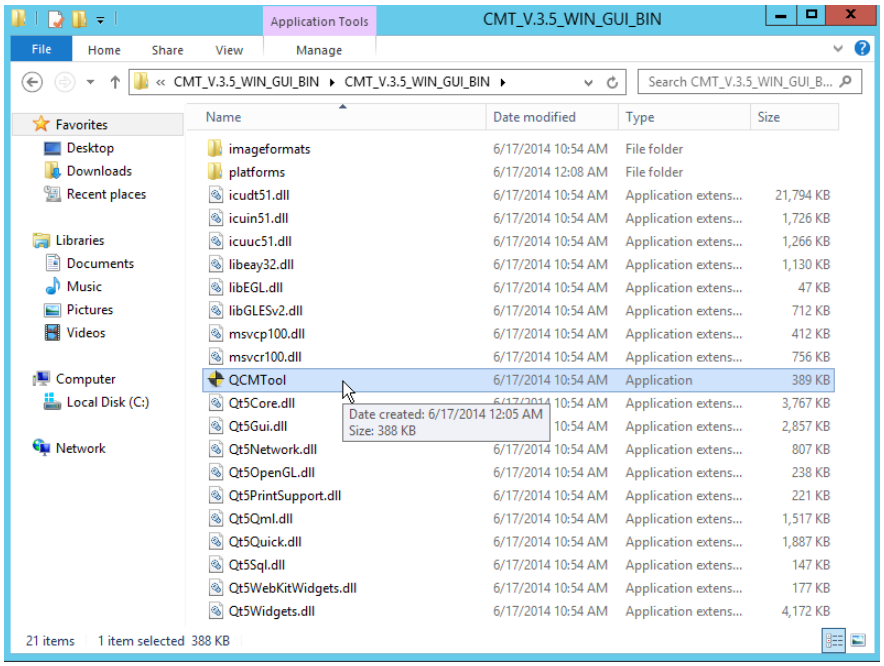
For Microsoft Windows-like operating systems, *CMT* is shipped as an executable application, so you do not actually need to install it.

Installing *CMT* on Windows Server 2012

1. Simply download *CMT* from QoP-ML's webpage.
2. Extract *CMT's* archive:



3. Run *CMT* by double-clicking *CMTool's* icon:



3 CMTool's Screenshoots

CMTool is available for GNU/Linux and Microsoft Windows operating systems. Supported operating system list includes (but it is not limited to):

- Ubuntu, Kubuntu, Xubuntu, Lubuntu, Debian, ...
- Linux Mint
- Fedora
- OpenSUSE
- CentOS
- Microsoft Windows XP
- Microsoft Windows Vista
- Microsoft Windows 7
- Microsoft Windows 8

Below are some screenshots of the *CMTool*. One can see how *CMTool's* interface looks on supported operating systems.

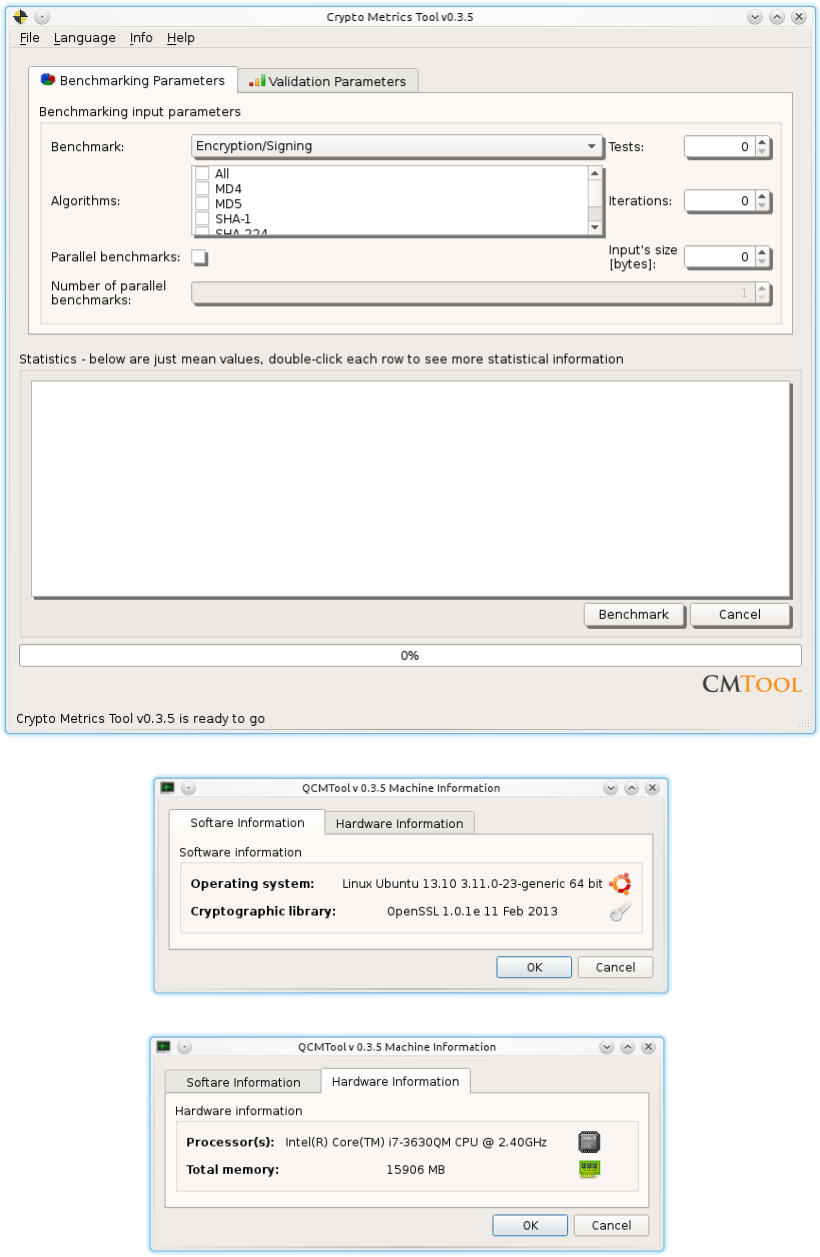


Figure 3.1: *CMTool* on Kubuntu 13.10 4 LTS, 64 bit.

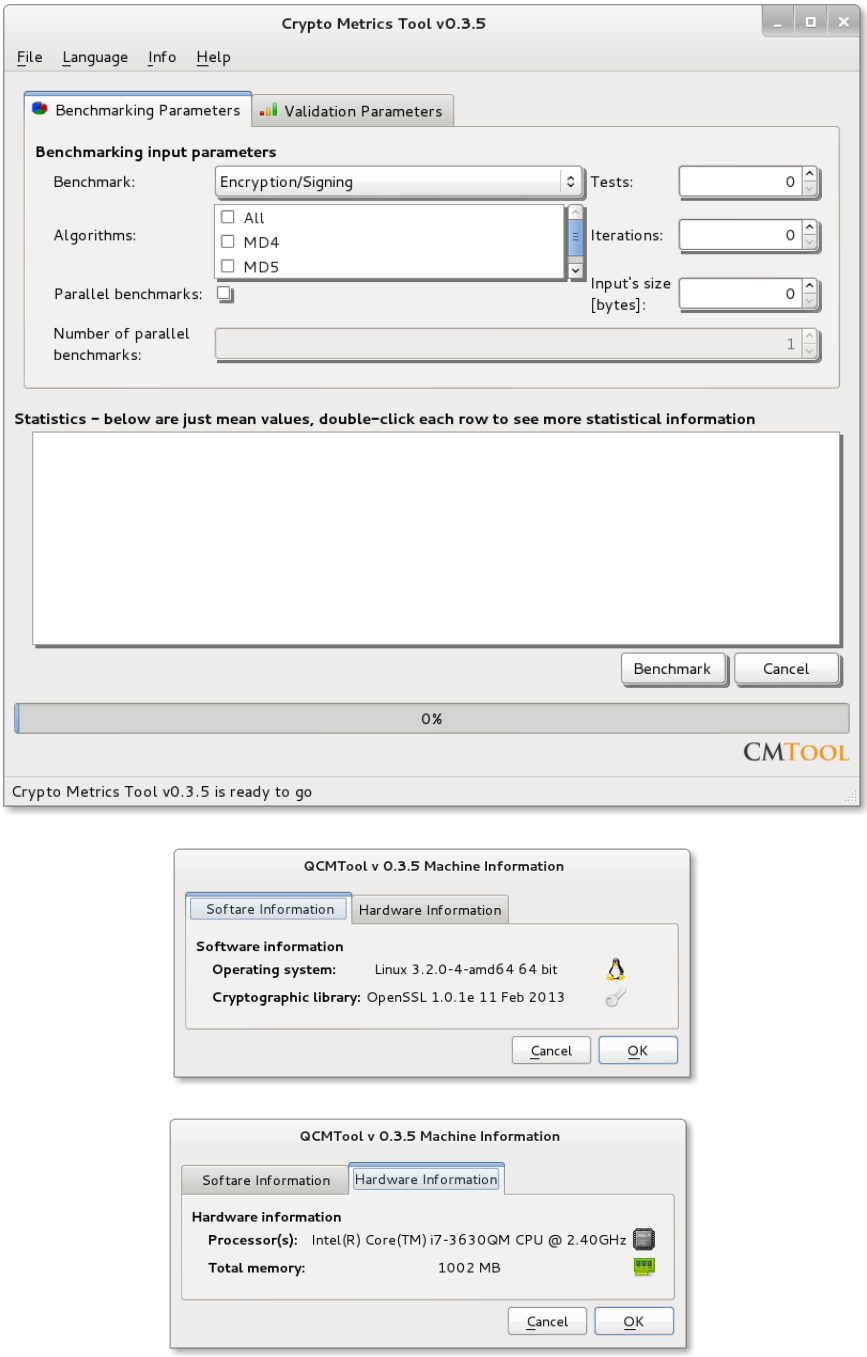
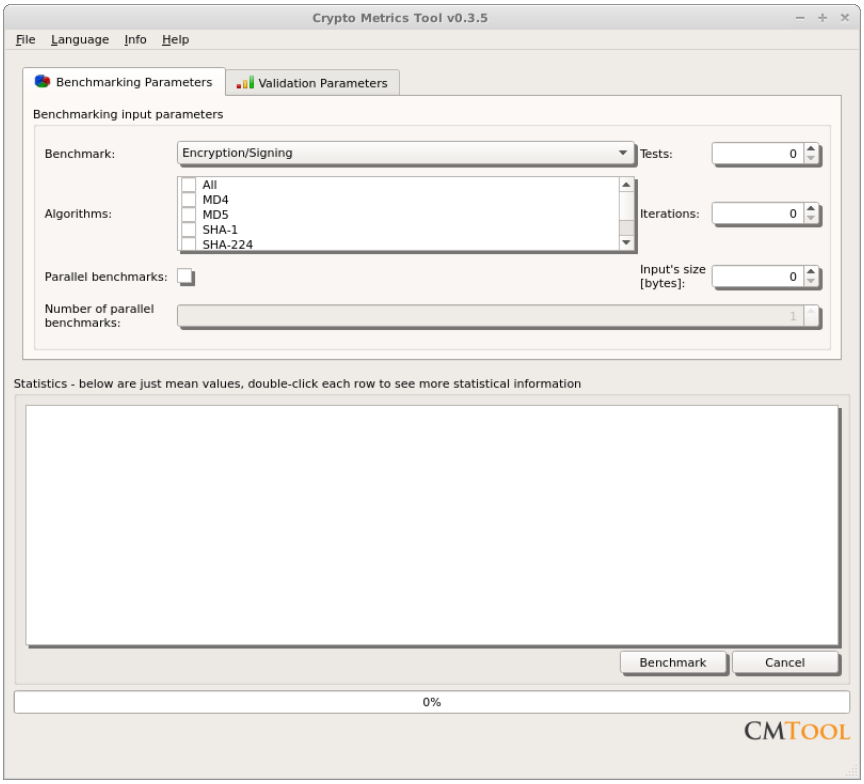
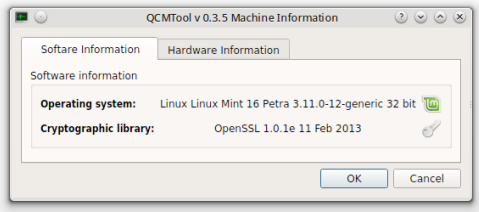


Figure 3.2: *CMTool* on Debian Wheezy 7, 32 bit.





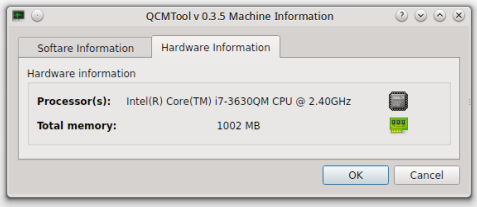


Figure 3.3: *CMTool* on Linux Mint 16 32 bit.

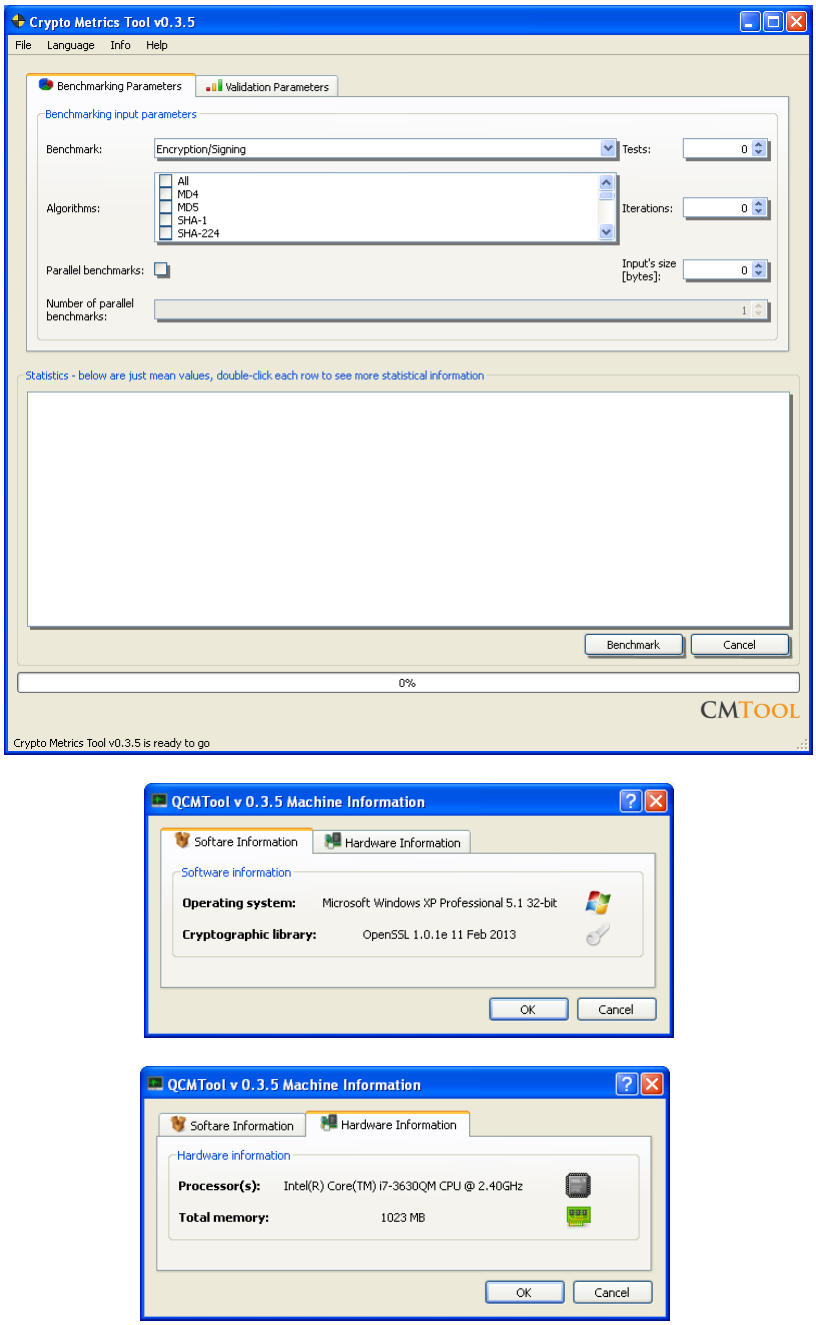


Figure 3.4: *CMTool* on Microsoft Windows XP, 32 bit.

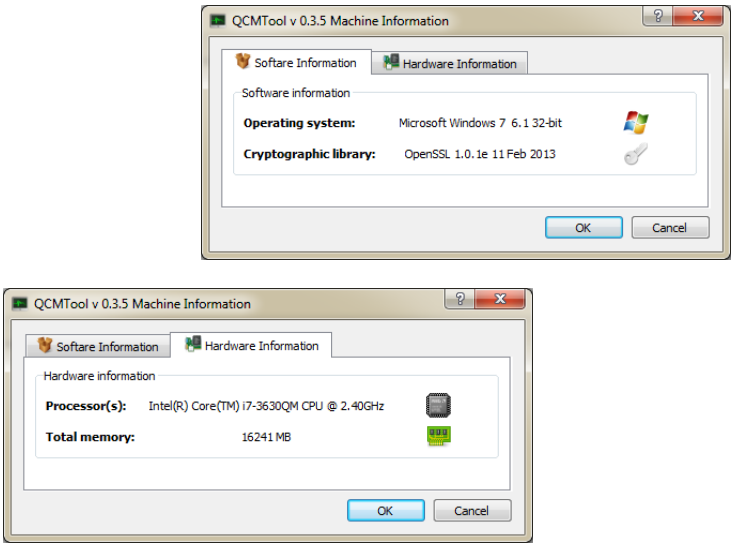


Figure 3.5: *CMTool* on Microsoft Windows 7, 32 bit.