

Decryption Guide for TeslaCrypt Encrypted Files

How is it possible to decrypt the encrypted files?

TeslaCrypt 2.2.0 and older uses the ECDH password exchange algorithm to protect the private key. The TeslaCrypt developers decided to multiply the calculated ECDH shared secret with the private key generated on a victim's computer. This decision allows us the possibility of recovering the original private key from $\text{SharedSecret} * \text{PrivateKey}$ by factoring this number to primes and then using these prime factors to reconstruct the original private key. Using various tools by BloodDolly and factoring utilities we can retrieve the information needed to recover and recreate the private key. Once the private key has been recovered, we can use TeslaDecoder to decrypt encrypted files.

It should be noted that determining the decryption key for a encrypted file could be very short (under 5 minutes) or very long (a couple of days). Therefore, the more powerful of a computer, the faster this process will take. Unfortunately, there is no way of determining how quick it will be to recover the decryption key.

Which versions can be decrypted by this method?

All versions from 0.3.4a to 2.2.0.

Extensions of compatible encrypted files are: ecc (0.3.4a+), ezz, exx, xyz, zzz, aaa, abc, ccc, vvv

TeslaCrypt 3.0.0 and newer cannot be decrypted by this method.

Extensions of incompatible encrypted files are: xxx, ttt, micro

Terminology used in this document:

My terminology may be different than what other search researchers use. To clear this up, I have defined below what each term means that is used in this process.

ECDH SharedSecret - [Elliptic curve Diffie–Hellman](#)

TeslaPrivateKey - Private key of Tesla's creators. There are 3 keys used in TeslaCrypt so far. These keys can decrypt every file encrypted by corresponding version of TeslaCrypt. These keys are unknown for me.

PrivateKeyBC - Private key used for bitcoin calculation. This key is used by TeslaCrypt as master key for infected computer. All files encrypted by this computer can be decrypted by this number. The main goal is to find this number.

PrivateKeySHA256BC - Hashed PrivateKeyBC using SHA256 algorithm

PublicKeyBC - Public key of calculated bitcoin address. More about bitcoin calculation can be found [here](#).

SharedSecretBC - Shared secret computed from PrivateKeyBC, PublicKeyBC and TeslaPublicKey. I named it SharedSecert1 in TeslaViewer.

PrivateKeyFile - This number is directly used as AES key for file encryption/ecryption. This number is changed everytime TeslaCrypt is executed on infected computer. This number can decrypt files encrypted

only in "current" session. If you are not able to get PrivateKeyBC in reasonable time you can try to use SharedSecret2*PrivateKeyFile and PublicKeyFile to recover this key.

PublicKeyFile - Public key of PrivateKeyFile.

SharedSecretFile - Shared secret computed from PrivateKeyFile, PublicKeyFile and PublicKeyBC or PublicKeySHA256BC. I named it SharedSecret2 in TeslaViewer.

Instructions:

In order to recover the decryption key for your files, we need to recover the PrivateKeyBC or PrivateKeyFile, but we first need to determine the corresponding shared secrets and public keys. This information can be obtained from one of the following sources:

- Recovery file (RECOVERY_KEY.TXT, RECOVERY_FILE.TXT, recovery_file_*.txt, recover_file_*.txt located in your documents folder)
- Any encrypted file
- Tesla's data file (key.dat, storage.bin)

Please note that this is a long and detailed guide. If you have any questions on how to use this tool, feel free to ask in the [TeslaDecoder topic](#).

Step 1: Setting up the Work Folder

The first step is to create a folder on your desktop called TD (for tesladecrypt) that we will use as our work folder. Then find one encrypted file and copy it into that folder. A good place to find an encrypted file is in the My Pictures\Samples folder as the sample images should all be encrypted.

Now download the following files into the TD folder.

- [TeslaDecoder](#) (Download again to make sure you have the latest version)
- [Yafu](#)

Step 2: Using TeslaViewer to extract SharedSecret1*PrivateKeyBC and PublicKeyBC.

Once the files are downloaded, go into the TD\TeslaDecoder folder and double-click on TeslaViewer. This will open a screen similar to the one below.

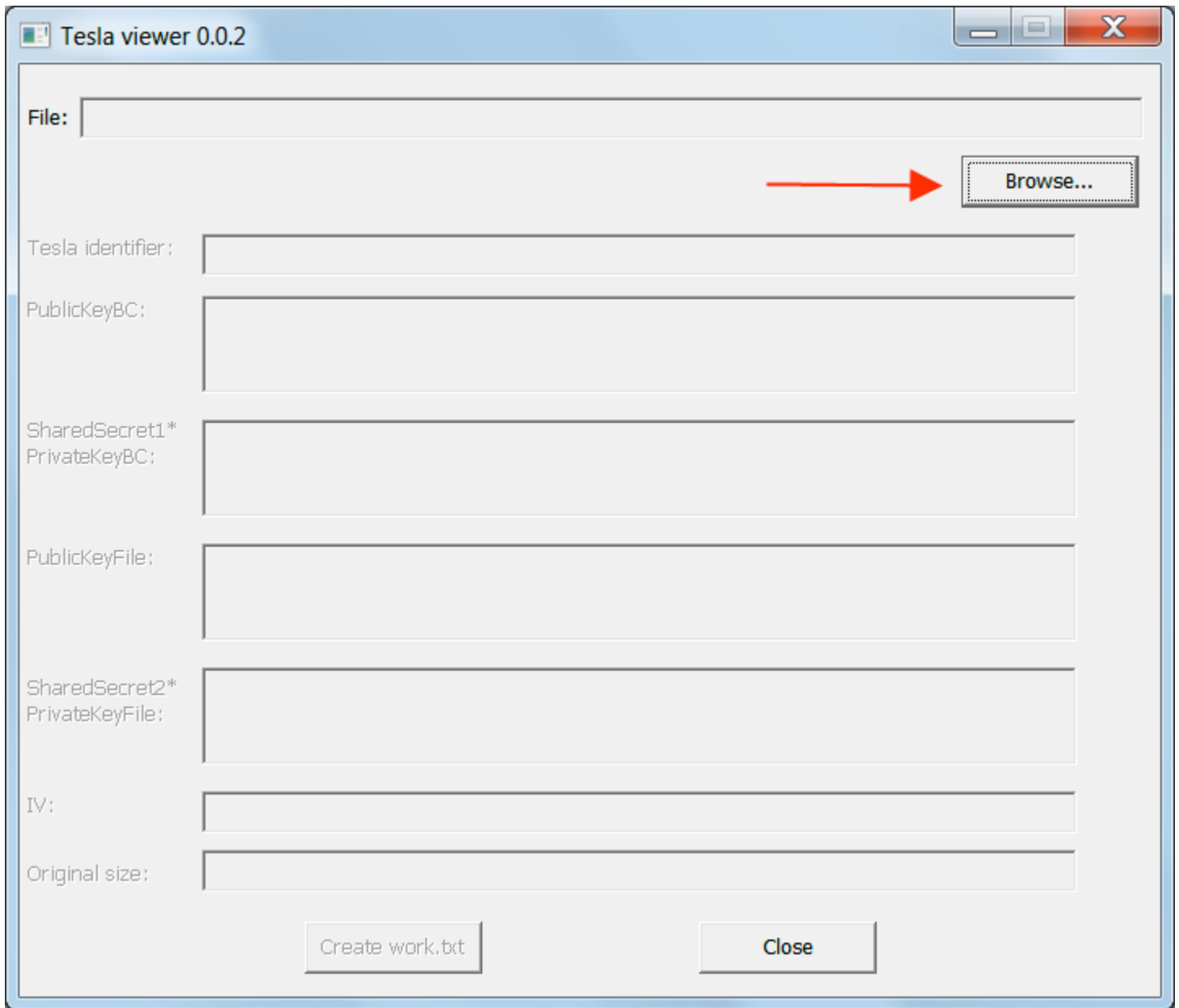


Figure 1. TeslaViewer

Click on the browse button as indicated by the red arrow and navigate to the TD folder that you saved your sample encrypted file. Then select the sample encrypted file that we will try to decrypt. TeslaViewer will now display a variety of information that was retrieved from the file as shown below.

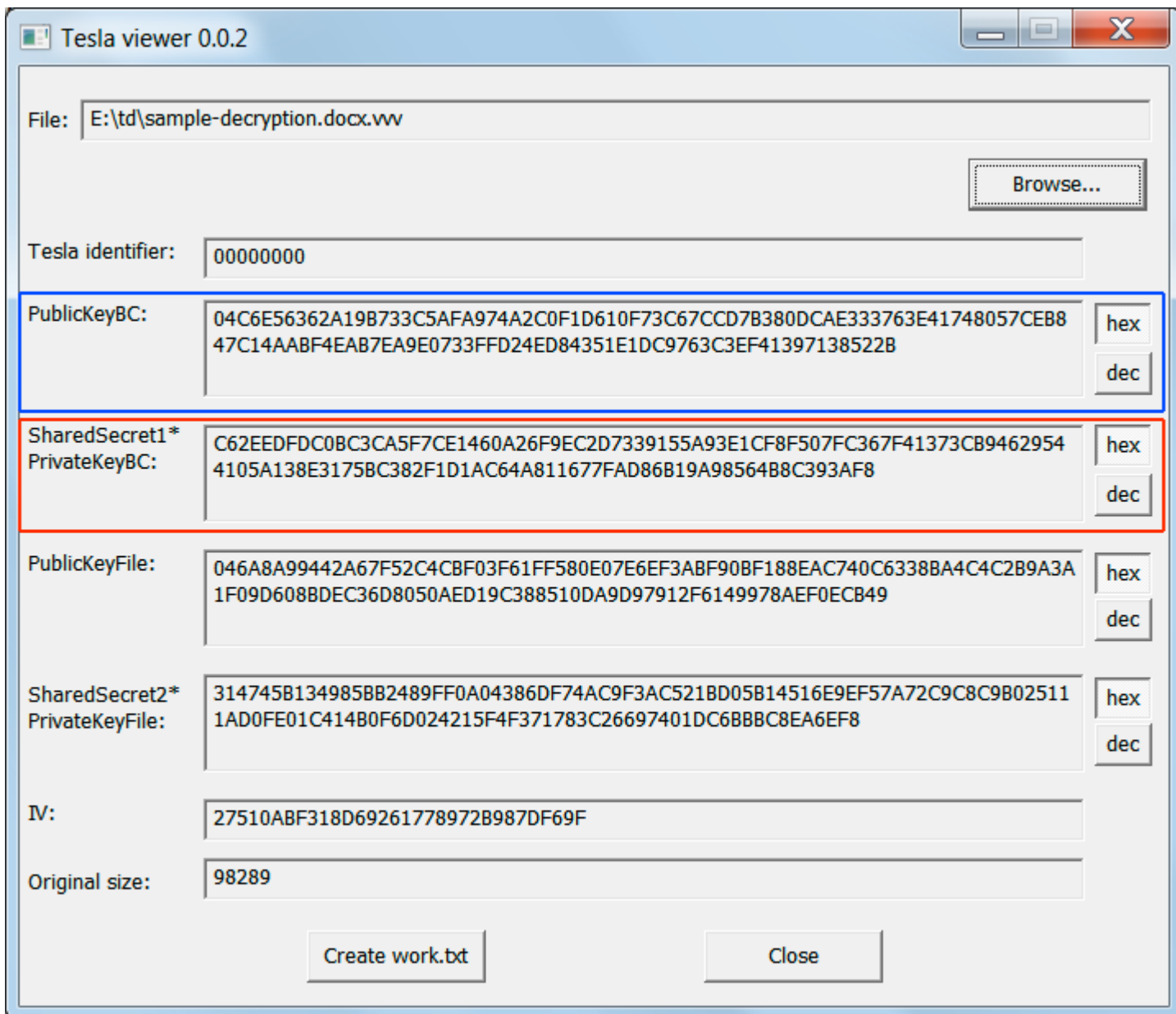


Figure 2. TeslaViewer Information

As you can see TeslaViewer recovered the information on the **PublicKeyBC** and the **SharedSecret1*PrivateKeyBC** that we need. To use this information, though, it is easier to copy it into a text file that we can copy from. To do this, click on the **Create work.txt** button and this information will be created in a file called work.txt located in the same location that TeslaViewer was run from.

An example of the saved work.txt file is shown below.

```
work.txt - Notepad2
File Edit View Settings ?
1 =====
2 = PrivateKeyBC =
3 =====
4
5 SharedSecret1*PrivateKeyBC
6 hex
  C62EEDFDC0BC3CA5F7CE1460A26F9EC2D7339155A93E1CF8F507FC367F41373CB94629544105A138E3175BC382F1
  D1AC64A811677FAD86B19A98564B8C393AF8
7 dec
  10379702638693384762811992165525587134314460673958666650723125787969707988300986376344309758
  165737840758089627026182763148898300015632008963640268229786360
8
9
10
11 PrivateKeyBC =
12 PublicKeyBC =
  04C6E56362A19B733C5AFA974A2C0F1D610F73C67CCD7B380DCAE333763E41748057CEB847C14AABF4EAB7EA9E07
  33FFD24ED84351E1DC9763C3EF41397138522B
13
14
15
16
17 =====
Ln 1:33 Col 1 Sel 0      1.09 KB      ANSI      CR+LF INS  Default Text
```

Figure 3. Work.txt

Now that we have the information need to start cracking the encryption, we move on to step 3 below.

Step 3: Saving time by seeing if the SharedSecret1*PrivateKeyBC number is factored already on Factordb.com

In order to recreate your decryption key, you need to perform prime factorization on the **SharedSecret1*PrivateKeyBC** decimal number. Unfortunately, prime factorization of large numbers is a complicated and potentially very long task. Thankfully there is a site called factordb.com that allows you to enter a number and see if it has been factored into its prime factors. As Factordb.com only accepts decimal forms of the numbers, we will need to copy the decimal number for **SharedSecret1*PrivateKeyBC** from the work.txt file that we created in the previous step. Below is the same work.txt file as above, but with the decimal number of **SharedSecret1*PrivateKeyBC** that we need to factor highlighted by the blue box.

```
1 =====
2 = PrivateKeyBC =
3 =====
4
5 SharedSecret1*PrivateKeyBC
6 hex
  C62EEDFDC0BC3CA5F7CE1460A26F9EC2D7339155A93E1CF8F507FC367F41373CB94629544105A138E3175BC382F1
  D1AC64A811677FAD86B19A98564B8C393AF8
7 dec
  10379702638693384762811992165525587134314460673958666650723125787969707988300986376344309758
  165737840758089627026182763148898300015632008963640268229786360
8
9
10
11 PrivateKeyBC =
12 PublicKeyBC =
  04C6E56362A19B733C5AFA974A2C0F1D610F73C67CCD7B380DCAE333763E41748057CEB847C14AABF4EAB7EA9E07
  33FFD24ED84351E1DC9763C3EF41397138522B
13
14
15
16
17 =====
```

Ln 1: 33 Col 1 Sel 0 1.09 KB ANSI CR+LF INS Default Text

Figure 4. Decimal SharedSecret1*PrivateKeyBC in Work.txt

Go to factordb.com and paste this number into the search field. When the results appear, you want to pay special attention to the **status** column. If the status column states, **FF** as shown below, you are very lucky as that means that the prime number is fully factored and you can skip the long task of factoring the numbers yourself. If it states **CF**, then only some of the factors are known and you will need to perform prime factorization on the number.

http://www.factordb.com

1037970263869338476281199216552558713431446067395866665072312578796970798830098637 Factorize! (?)

Result:

status	digits	number
FF	155 (show)	<u>1037970263...60</u> _{<155>} = <u>2³</u> · <u>3</u> · <u>5</u> · <u>7</u> · <u>29</u> · <u>283</u> · <u>5441</u> · <u>23827</u> · <u>694407479587225887111618306307</u> _{<30>} · <u>65908736209941917092087881680509</u> _{<32>} · <u>26392081893794160933561951385008001</u> _{<35>} · <u>9614840347780751770439646130515390398878117</u> _{<43>}

[More information](#)

[ECM](#)

factordb.com - 44 queries to generate this page (0.24 seconds) ([limits](#)) ([Imprint](#))

Figure 5. Fully Factored Number

If it showed that it was fully factored (FF), then your job is essentially done. Simple copy each factor onto its own line, as shown by the underlined numbers above, into a file as we will need to use them in a later step. Sometimes the factors will be displayed as something like **1884516739...69**_{<146>}. In this situation, you need to click on the factor in order to see the full number, which would be 146 digits long in this example. Now that you have safely copied each factor into a file, with each factor on its own line, you can now [skip to Step 5](#) to rebuild your private decryption key.

Unfortunately, if it shows a status of **CF** as shown below, then it means only some factors are known and you still need to perform the prime factorization on your own.

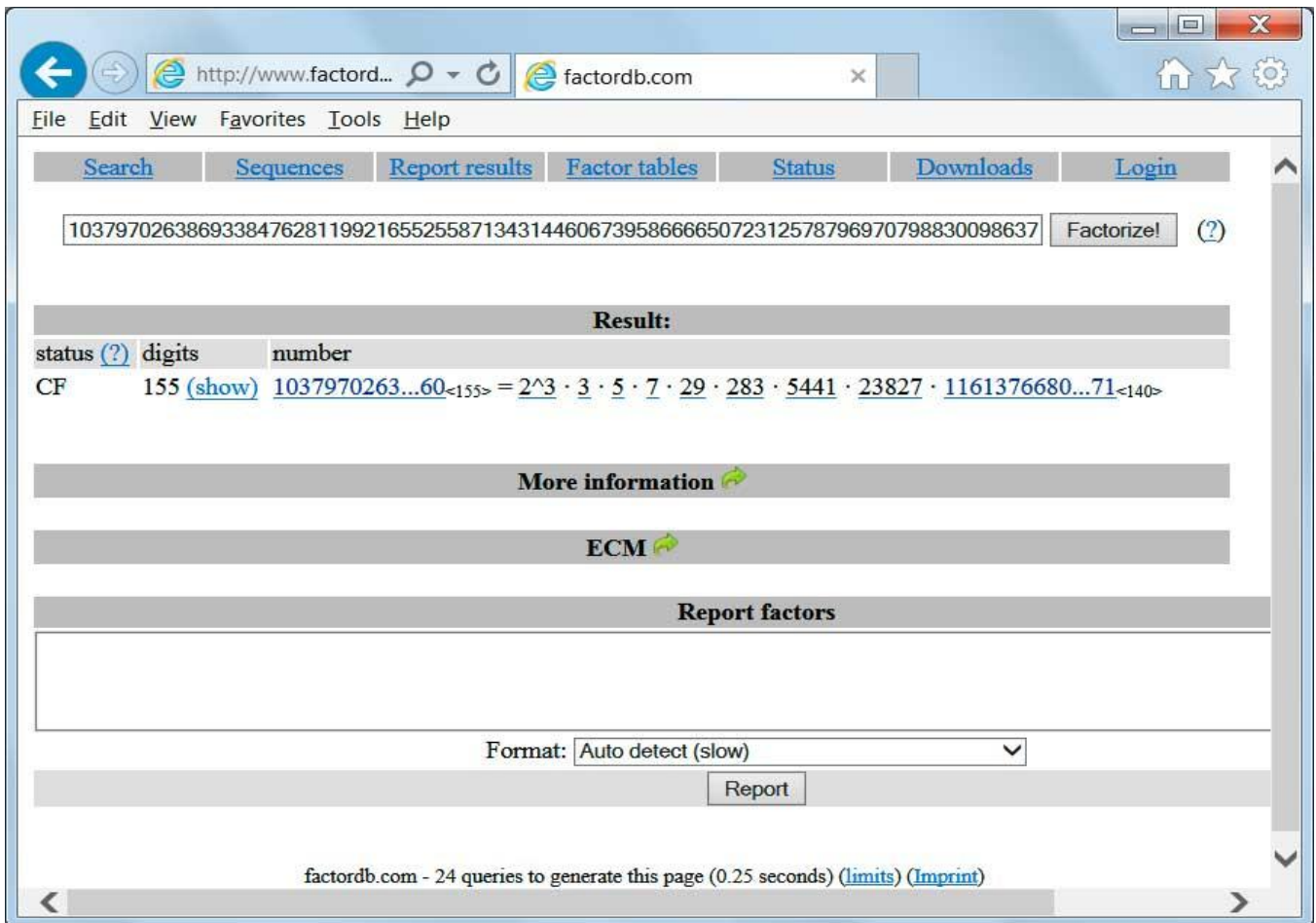


Figure 6. Not fully factored

Since your number is not already factored, you will need to perform prime factorization on it using the step below.

Step 4: Prime factorization of the SharedSecret1*PrivateKeyBC decimal number

In order to perform prime factorization on the **SharedSecret1*PrivateKeyBC** we need to use specialized tools such as Yafu and Msieve. In this guide we are going to use the Yafu tool that you downloaded [Step 1](#). Yafu can normally be found [here](#), but that version requires you to download various other components such as [GGNFS](#) and [GMP-ECM](#) to get it to work properly. To make it easier we have created a Yafu download with the necessary components already bundled. They can be downloaded again here if you did not do it in [Step 1](#).

[Yafu Download Link](#)

Below are instructions on how to factor your **SharedSecret1*PrivateKeyBC** decimal number using Yafu.

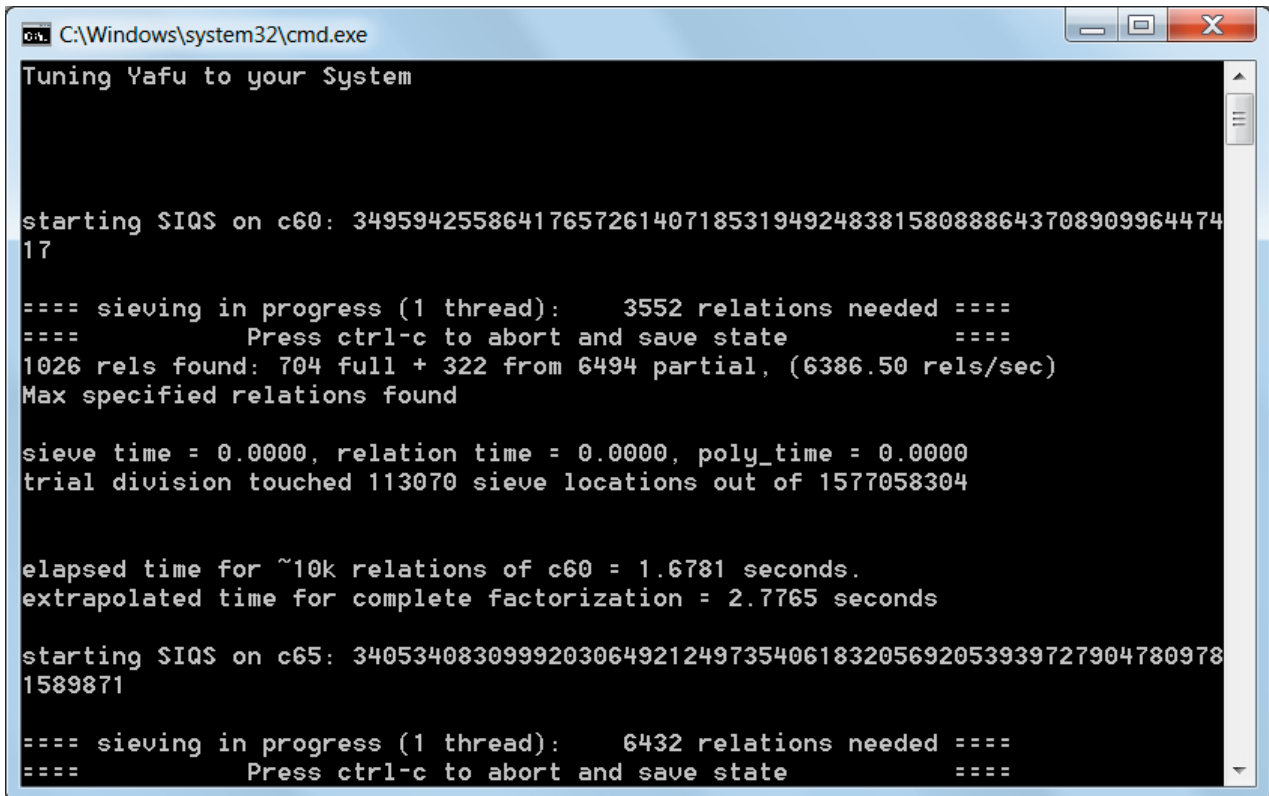
For those who are unfamiliar with the Windows command line, we have created some batch files that make it easier to perform this process if you are not familiar with the command line. Each batch file has a x86 and x64 version that should be used depending on what bit-type your OS is. You can determine what bit-type your computer is by [reading this tutorial](#).

Before factoring your number, you should double-click on the **tuneX86.bat** (32-bit users) or the **tuneX64.bat** (64-bit users) batch file to optimize Yafu for your computer. This process can take a while, so please be patient while it completes. You can also use the command line to tune Yafu by using the appropriate command below if you do not wish to use the tune.bat batch file:

Yafu x86: yafu-Win32.exe tune()

Yafu X64: yafu-x64.exe tune()

When the tuning process starts you will see a window similar to the one below.



```
C:\Windows\system32\cmd.exe
Tuning Yafu to your System

starting SIQS on c60: 3495942558641765726140718531949248381580888643708909964474
17

==== sieving in progress (1 thread):    3552 relations needed ====
====          Press ctrl-c to abort and save state          ====
1026 rels found: 704 full + 322 from 6494 partial, (6386.50 rels/sec)
Max specified relations found

sieve time = 0.0000, relation time = 0.0000, poly_time = 0.0000
trial division touched 113070 sieve locations out of 1577058304

elapsed time for ~10k relations of c60 = 1.6781 seconds.
extrapolated time for complete factorization = 2.7765 seconds

starting SIQS on c65: 3405340830999203064921249735406183205692053939727904780978
1589871

==== sieving in progress (1 thread):    6432 relations needed ====
====          Press ctrl-c to abort and save state          ====
```

Figure 7. Tuning

When you are done, you can now begin to factor your number using Yafu. Open work.txt and copy the decimal number associated with **SharedSecret*PrivateKeyBC** as shown in Figure 4 above. To start factoring, click on the **factorX86.bat** (32-bit users) or the **factorX64.bat** (64-bit users) batch file. When the script asks you for the number you wish to factor, you should copy the decimal number from your work.txt and then click on the black factor.bat window. Once you have clicked on the factor.bat window, right click with the mouse button to paste the number into the window. Once the number has been pasted you can press **enter**.

The batch file will then ask you for the amount of threads you wish to use. You should typically set this number to be the amount of logical CPUs minus 1, so that you have one CPU left over to do other work. You can see how many logical CPUs you have by going into Task Manager and then clicking on the Performance as shown below.

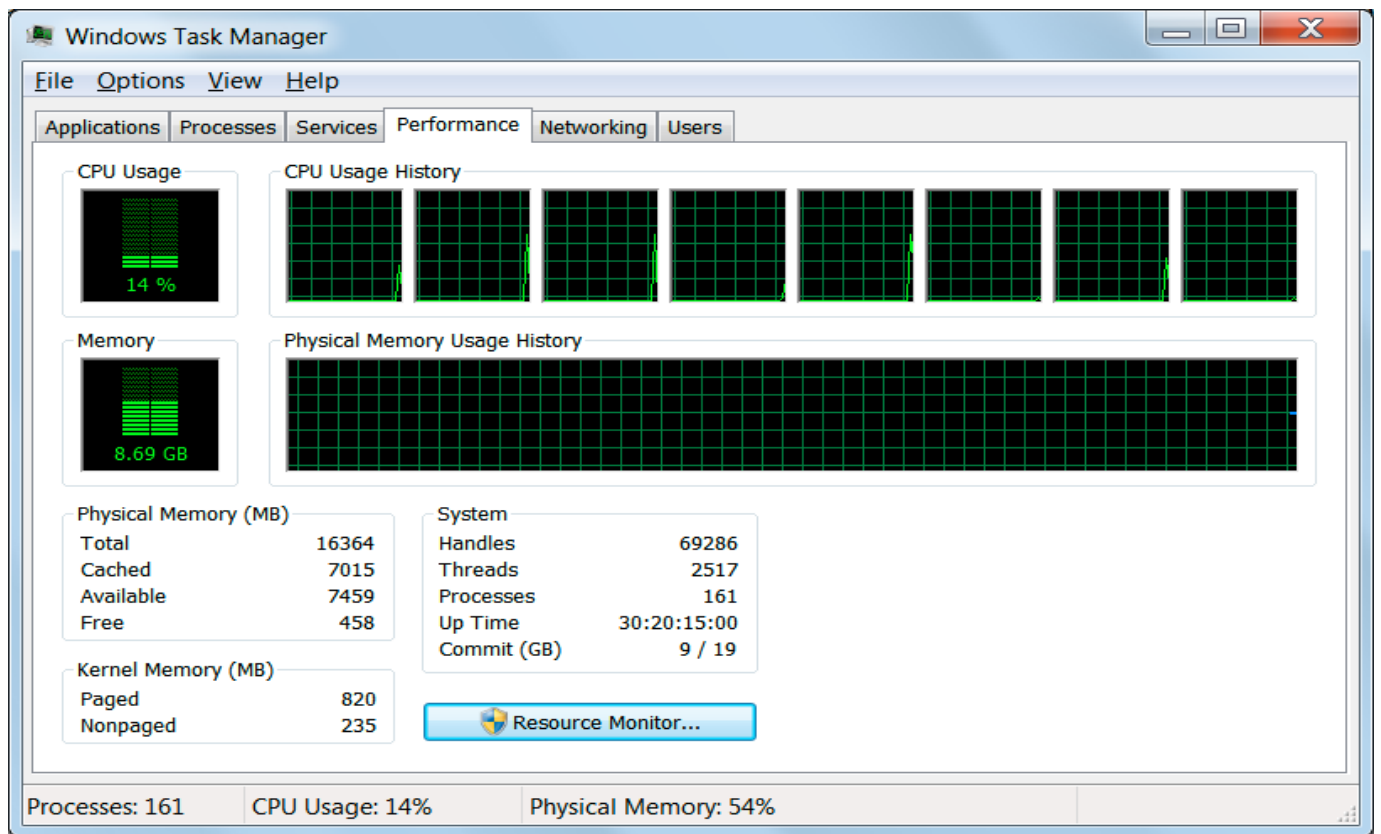


Figure 8. Logical CPUs

Notice how in the Figure 8 above it shows that I have 8 logical CPUs. So, you would 7 in the number of threads to use and press **enter**. Once you enter the number of threads and press enter, Yafu will start factoring your number.

You can also use the command line to factor the number using the commands below:

Yafu x86: `yafu-Win32.exe factor(SharedSecret1*PrivateKeyBC) -threads %numthreads% -ecm_path .\ecm70dev-svn2256-win32-gc\ecm.exe -ggnfs_dir .\ggnfs-svn413-win32-p4\`

Example: `yafu-Win32.exe factor(1037970263869338476281199216552558713431446067) -threads 7 -ecm_path .\ecm70dev-svn2256-win32-gc\ecm.exe -ggnfs_dir .\ggnfs-svn413-win32-p4\`

Yafu X64: `yafu-x64.exe factor(SharedSecret1*PrivateKeyBC) -threads %numthreads% -ecm_path .\ecm644_win64_i7\ecm.exe -ggnfs_dir .\ggnfs-svn413-win64-core2\`

Example: `yafu-x64.exe factor(1037970263869338476281199216552558713431446067) -threads 7 -ecm_path .\ecm644_win64_i7\ecm.exe -ggnfs_dir .\ggnfs-svn413-win64-core2\`

The factoring of your **SharedSecret1*PrivateKeyBC** number will now begin and you will screen similar to the one below.

```
C:\Windows\system32\cmd.exe
Enter the decimal number for SharedSecret1*PrivateKeyBC that you retrieved from
TeslaView:
Enter DEC SharedSecret1*PrivateKeyBC:1037970263869338476281199216552558713431446
06739586666507231257879697079883009863763443097581657378407580896270261827631488
98300015632008963640268229786360

Enter the amount of threads you wish to use to crack the key.
You can determine the amount of threads by opening Task Manager and clicking on
the Performance tab.
In that tab will be the amount of CPUs available. I suggest you enter NumCPUs-1
as your thread amount.
Amount of Threads:7

fac: factoring 10379702638693384762811992165525587134314460673958666650723125787
96970798830098637634430975816573784075808962702618276314889830001563200896364026
8229786360
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
fmt: 1000000 iterations
rho: x^2 + 3, starting 1000 iterations on C144
rho: x^2 + 2, starting 1000 iterations on C144
rho: x^2 + 2, starting 1000 iterations on C140
```

Figure 9. Factoring

As already stated, this process can take a **long** time or a short time depending on the number, please be patient. When Yafu has finished it will display a list of prime factors as shown below.

```
C:\Windows\system32\cmd.exe
pm1: starting B1 = 15M, B2 = gmp-ecm default on C110
ecm: 343/616 curves on C110, B1=1M, B2=gmp-ecm default, ETA: 3.5 min
Total factoring time = 386.2482 seconds

***factors found***

P1 = 2
P1 = 2
P1 = 2
P1 = 3
P1 = 5
P1 = 7
P2 = 29
P3 = 283
P4 = 5441
P5 = 23827
P30 = 694407479587225887111618306307
P35 = 26392081893794160933561951385008001
P32 = 65908736209941917092087881680509
P43 = 9614840347780751770439646130515390398878117

ans = 1
Press any key to continue . . .
```

Figure 10. Factors Found

You now need to copy each of these factors into the work.txt so we can use it in the next step. When copying them, you should copy each one on its own line. So when done, the factors should be listed in work.txt as:

```
2
2
2
3
5
7
29
283
5441
23827
694407479587225887111618306307
65908736209941917092087881680509
9614840347780751770439646130515390398878117
26392081893794160933561951385008001
```

You can now proceed to step 5 below to recreate your decryption key.

Step 5: Refactoring the private key

Now that we have all the factors of the **SharedSecret1*PrivateKeyBC** number, we need to use TeslaRefactor to reconstruct the private decryption key. When you start TeslaRefactor you will be presented with a screen similar to the one below.

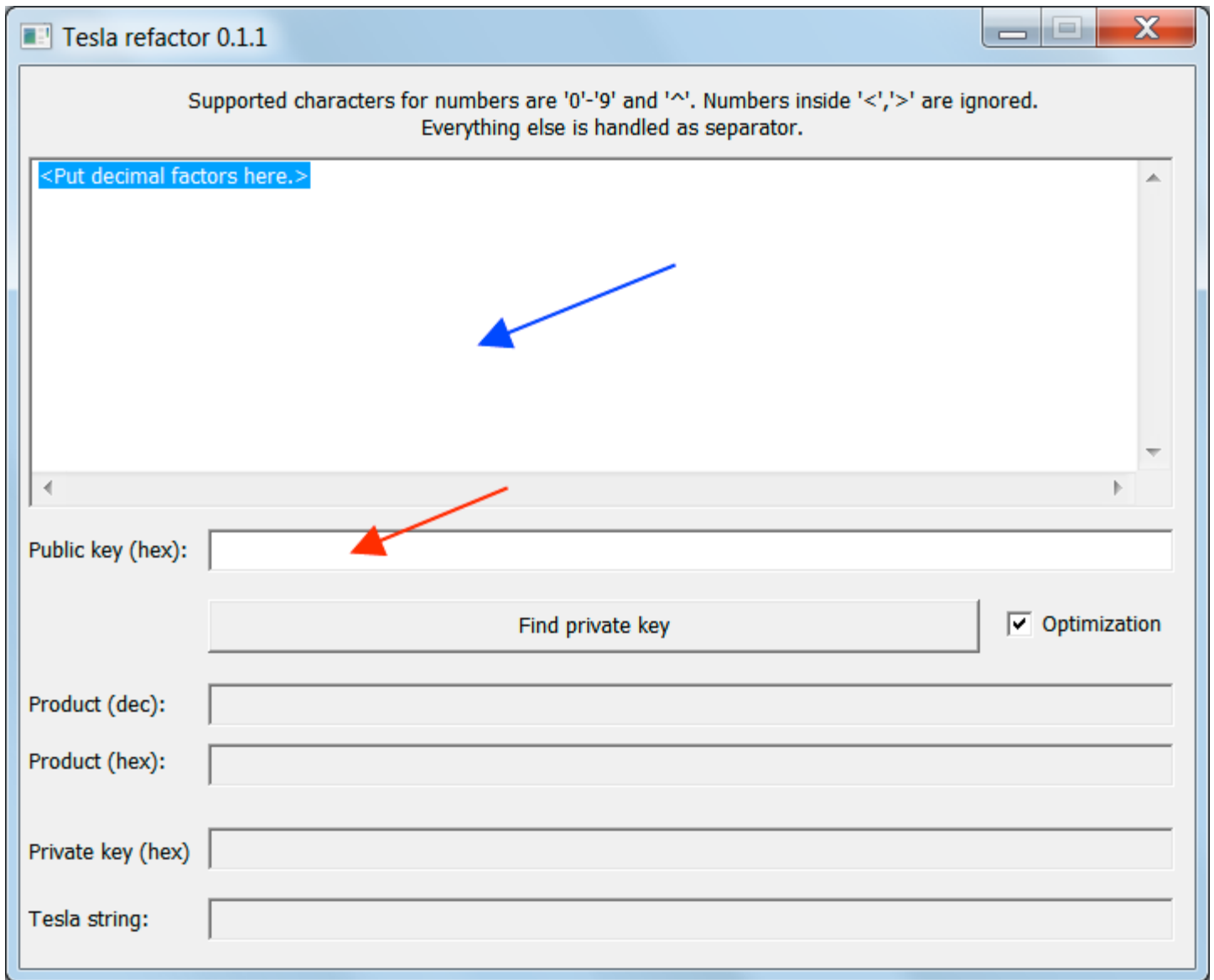


Figure 11. TeslaRefactor

You now need to copy the list of factors that you created in your work.txt document and paste them into the large text designated by the blue arrow. You then need to copy the **PublicKeyBC** found in work.txt and copy it into the **Public Key (hex)** field as designated by the red arrow.

Your TeslaRefactor screen will now look like the following.

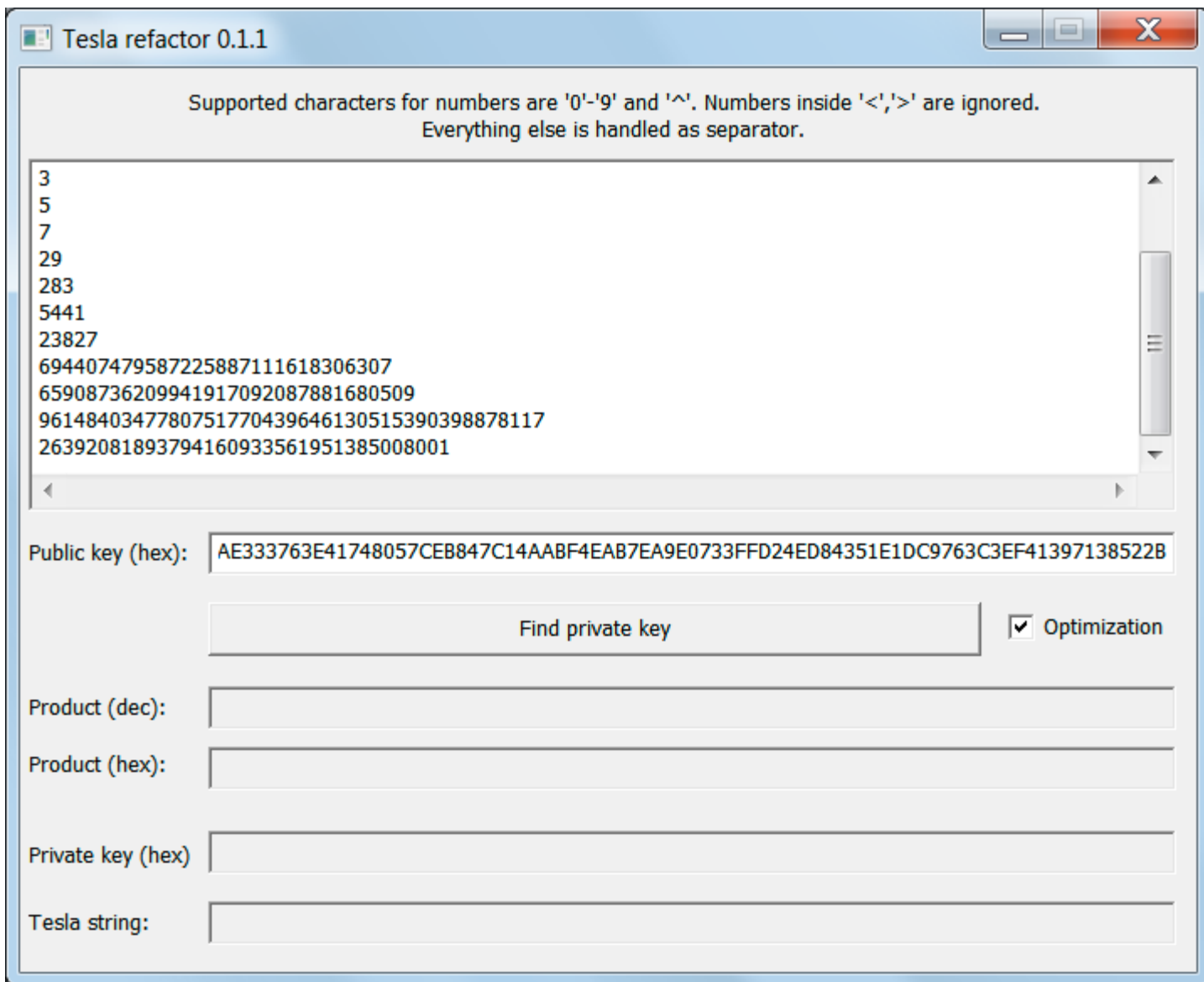


Figure 12. TeslaRefactor with Information Filled In

To reconstruct get your private decryption key, simply click on the **Find Private Key** button. TeslaRefactor will reconstruct the key and display it in the program as shown in Figure 13 below.

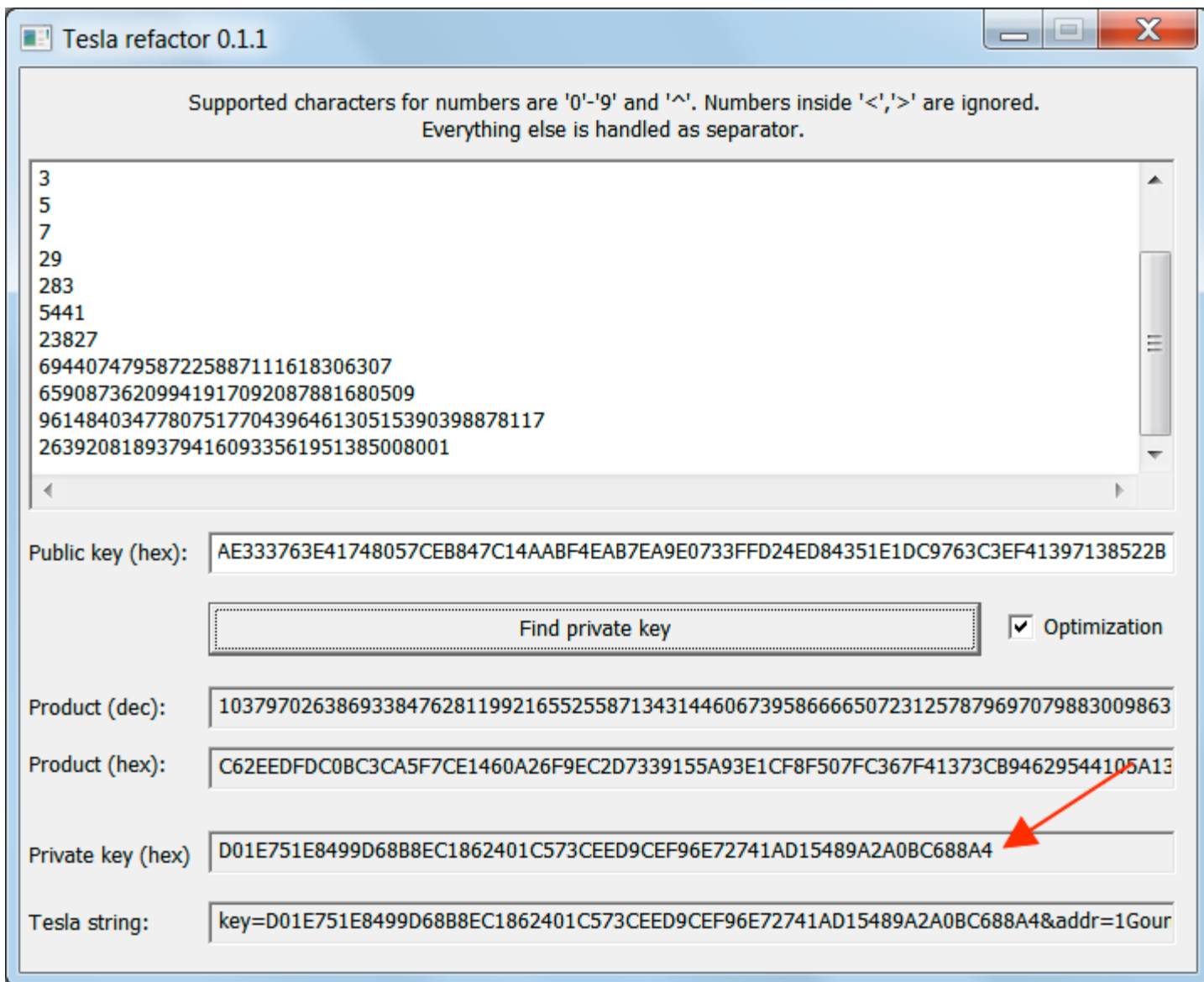


Figure 13. Found Private Key

As you can see, in the above image we found the private key

D01E751E8499D68B8EC1862401C573CEED9CEF96E72741AD15489A2A0BC688A4.

If there was a problem finding the private key, please check compare the value of the Product (dec) and Product (hex) with the SharedSecret1*PrivateKeyBC values in the work.txt file. If the product is the same as in the work.txt file, uncheck the **Optimization** checkbox and click on the **Find Private key button** again.

If the program is still unable to find the key, [contact BloodDolly](#) via PM on BleepingComputer.com.

If you have the Private Key, copy it to your work.txt file so we can use it in the next step.

Step 6: Decrypting your files with the refactored Private Key

Now that you have recovered the Private Key you can decrypt your files using TeslaDecoder. To do this, double-click on the TeslaDecoder.exe program so that the main screen is displayed as shown in Figure 14

below. To gain access to personal or hidden folders you need to execute TeslaDecoder.exe as administrator. To do so, right-click on TeslaDecoder.exe and click on Run as Administrator option.

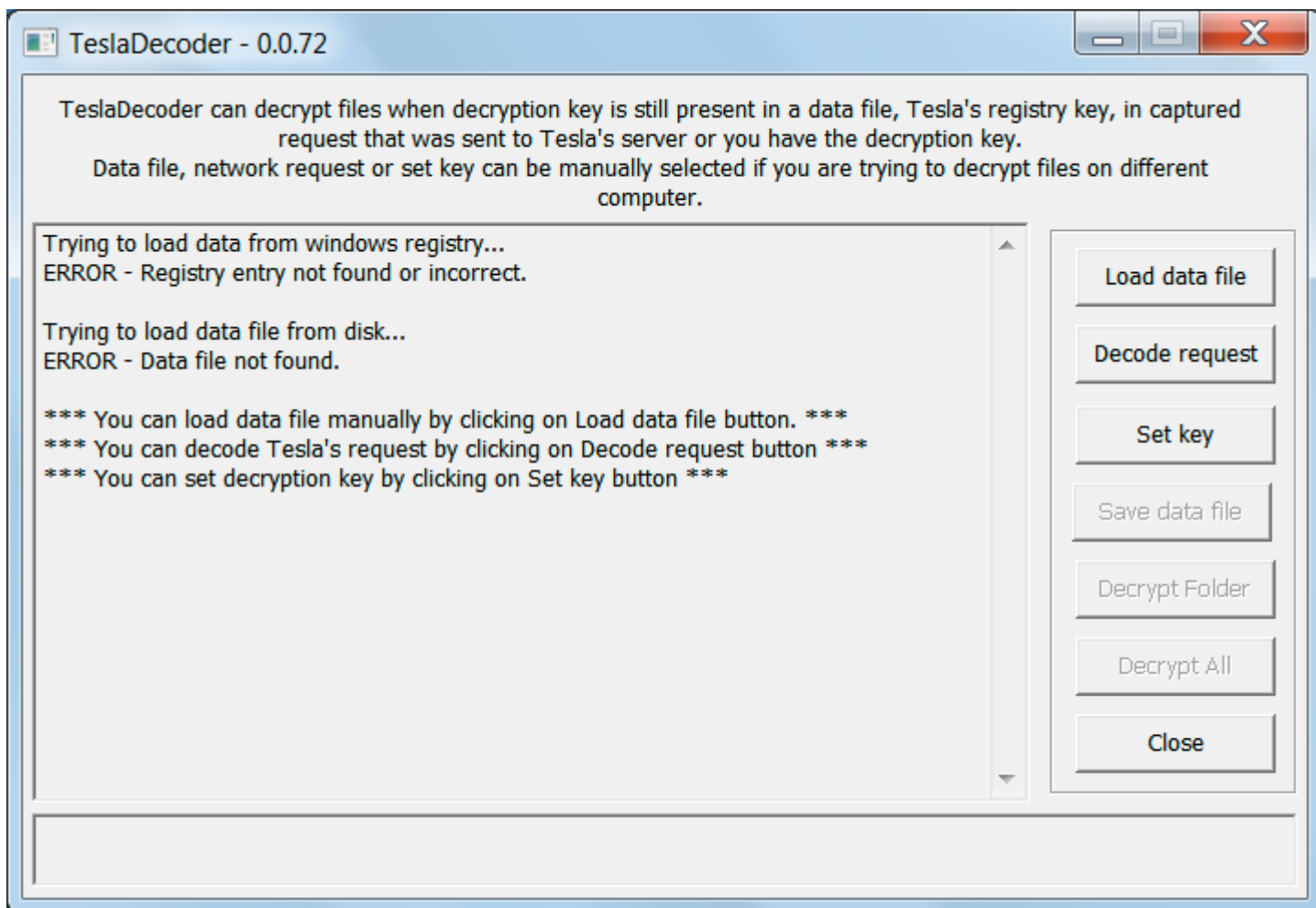


Figure 14. TeslaDecoder

When the main screen is shown, click on the **Set Key** button and paste the private key you recovered from the previous step. Then select the extension of your encrypted files. You can see an example of this in the image below.

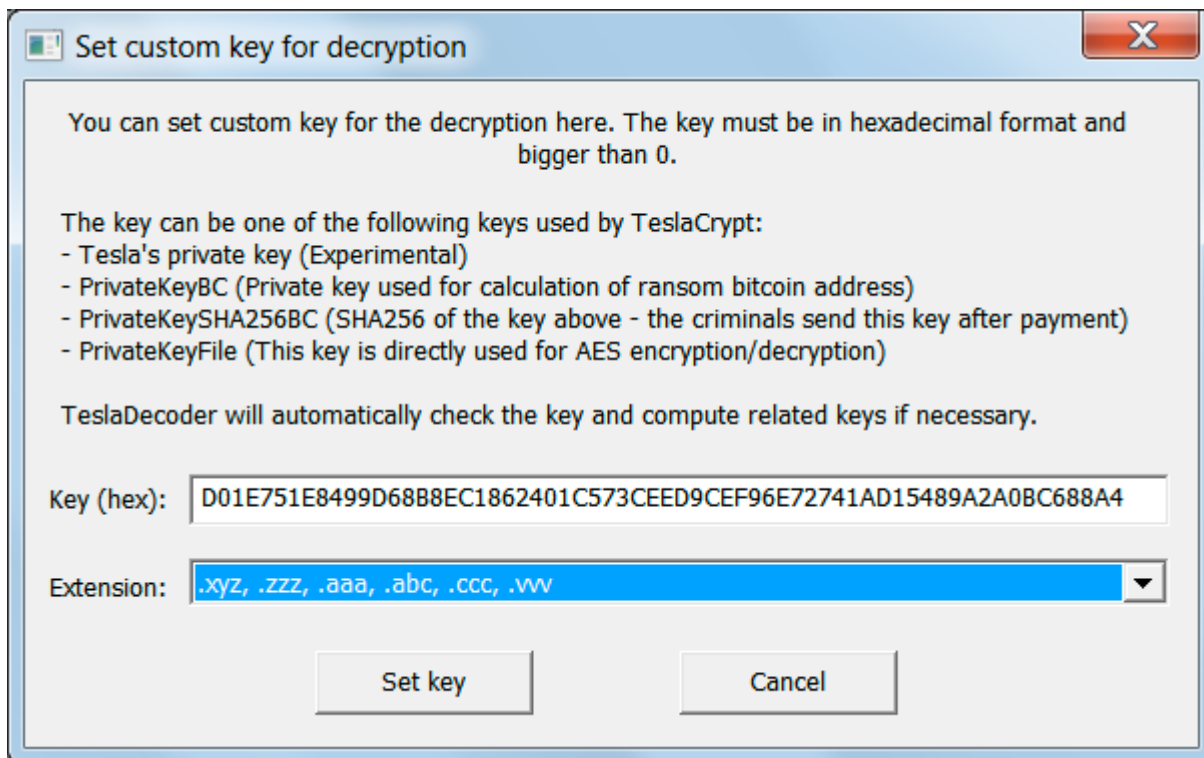


Figure 15. Setting the Private Key

Now click on the **Set key** button to load the key into TeslaDecoder. You can now perform a test decryption on the sample file we copied to the TD folder we created earlier. If the file is decrypted successfully, then use the **Decode All** button to decrypt all the files on your hard drive.

If you have any questions or need help using this guide, please ask in the [TeslaDecoder topic](#).

BloodDolly and Lawrence Abrams

© 2015-2016 *BloodDolly*

© 2015-2016 [BleepingComputer.com](#) All Rights Reserved.