Cryptully Documentation

Release 5.0.0

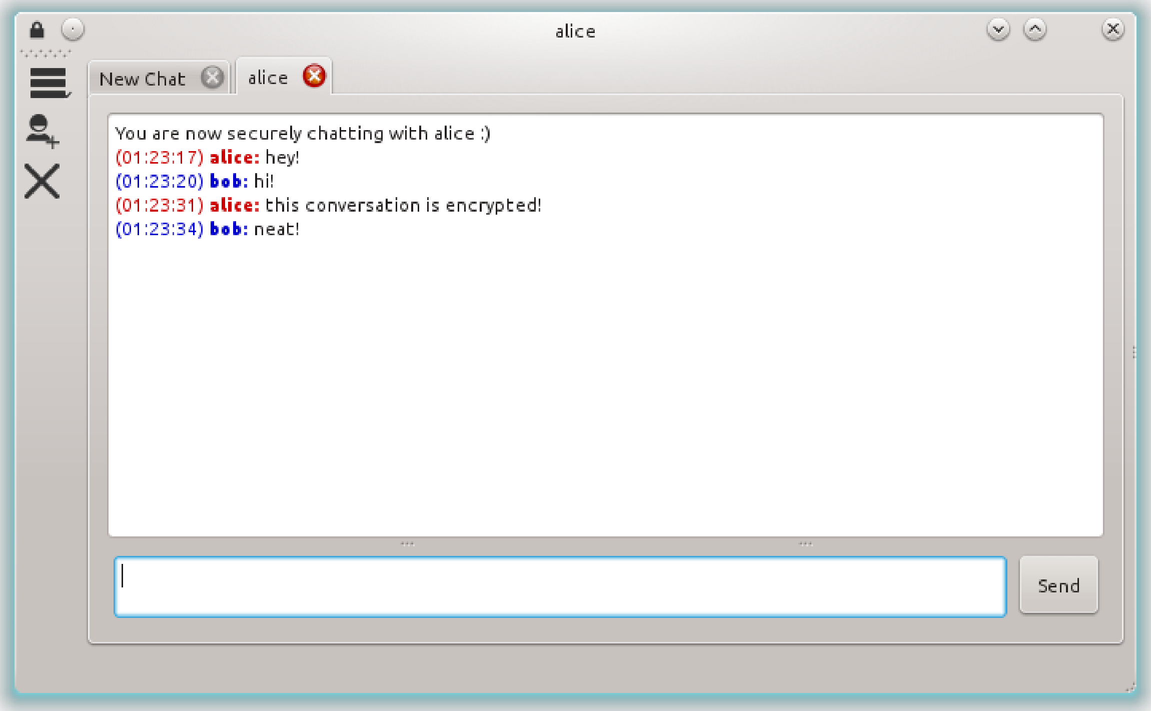
Shane Tully

November 15, 2015

CHAPTER 1

Introduction

Cryptully is an encrypted chat program meant for secure conversations between two people with no knowledge of cryptography needed.



```
You are now securely chatting with alice :)
(01:23:17) alice: hey!
(01:23:20) bob: hi!
(01:23:31) alice: this conversation is encrypted!
(01:23:34) bob: neat!
```

CHAPTER 2

Features

- Provides basic, encrypted chat with no prerequisite knowledge of cryptography
- Runs on Linux, Windows, and Mac OS X
- No registration or software installation required
- Chat with multiple people simultaneously
- Ability to host your own server (for the technically inclined)
- Graphical UI and command line (Curses) UI
- Open source (LGPL license)

2.1 How does it work and how is it secure?

Cryptully works by relaying messages from one person to another through a relay server. It generates per-session encryption keys (256bit AES) that all communications are encrypted with before leaving your computer and then decrypted on the destination computer.

CHAPTER 3

Quick Start

1. Download the executable for your platform on the *Downloads* page.
2. Launch the executable (no need to install anything).
3. Select a nickname and connect to the server.
4. Enter the nickname of the person you want to chat with.
5. You should now be chatting!

Need more info? See the *Using Cryptully* page for much more detailed instructions.

CHAPTER 4

Doesn’t encrypted chat already exist?

Yup, it does. There are plenty of other encrypted chat programs so what’s the point of Cryptully? The problem is just that, there’s plenty of other chat programs. There’s too many options for chating with another person. Other solutions require downloading and installing software, creating accounts, etc. With Cryptully, you just download and run the software. No need to install anything or create an account. Just enter the nickname of the person you want to chat with and you’re off.

Another advantage is that Cryptully is a relatively simple program and is open source. For the paranoid, you can inspect the source code to ensure that Cryptully is not doing anything nefarious or host your own relay server.

CHAPTER 5

Contents

5.1 Downloads

Cryptully is available for Linux, Windows, and OS X. See the releases page on GitHub for download links.

https://github.com/shanet/Cryptully/releases

5.2 Using Cryptully

5.2.1 Getting Cryptully

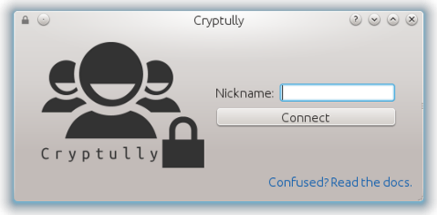
The first step is downloading Cryptully. To do that, head over to the *Downloads* page. Cryptully is available for Linux, Windows, and OS X. Just download the file and run it. No need to install anything or create any accounts.

5.2.2 Connecting to a friend

Cryptully uses a central server to relay messages from one person to another.

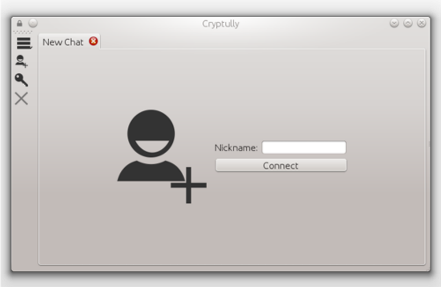
Let's run through the process of connecting with a friend.

1. When you first open Cryptully, you'll see the following screen:

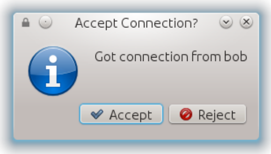


2. Pick a nickname that will identify you to other people you'll chat with.

3. Once connected to the server, you may enter the nickname of someone you wish to chat with.

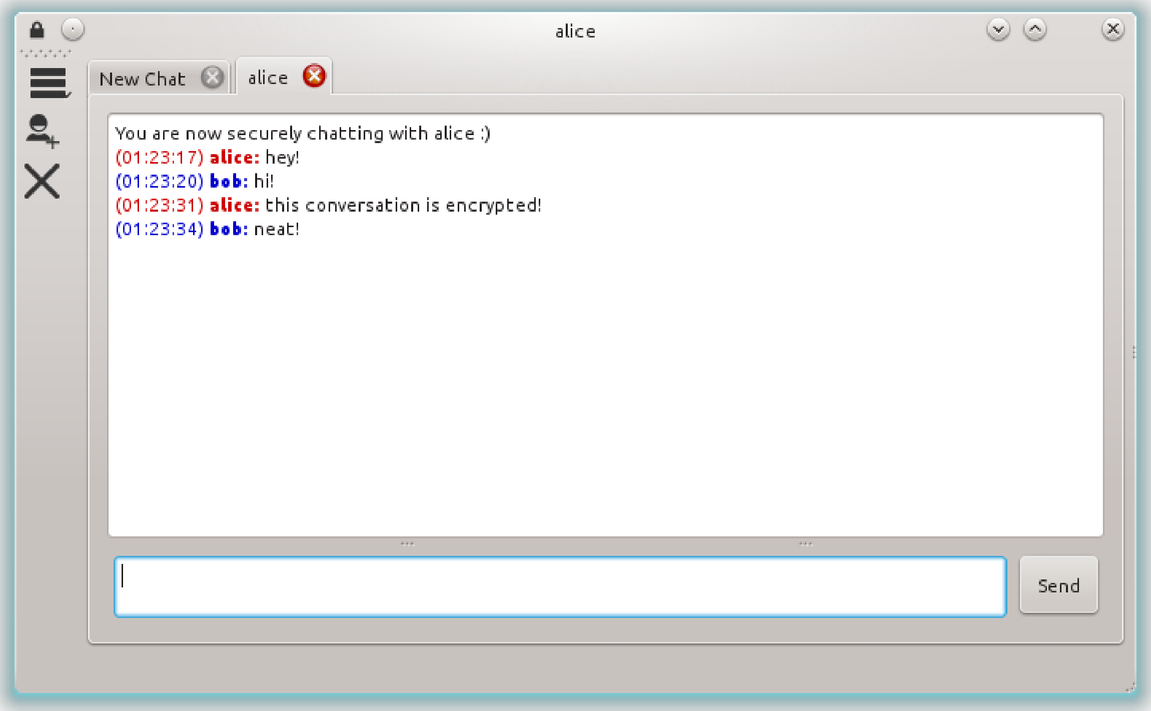


4. If the connection was successful, the person being connected to will see a dialog asking to accept or reject the connection.



Got connection from bob

5. Upon accepting the connection, both people are chatting securely!



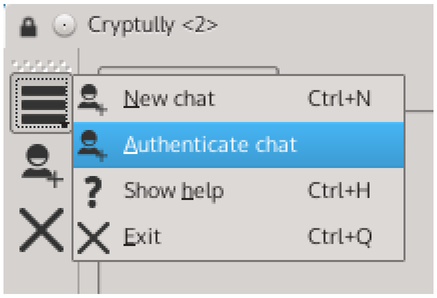
```
You are now securely chatting with alice :)
(01:23:17) alice: hey!
(01:23:20) bob: hi!
(01:23:31) alice: this conversation is encrypted!
(01:23:34) bob: neat!
```

5.2.3 Chat Authentication

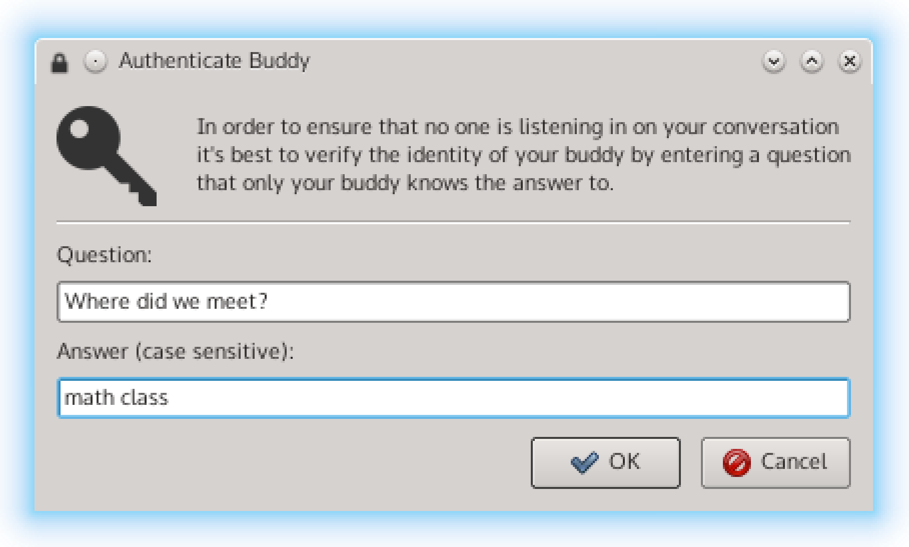
In order to verify the person you're chatting with is not an impersonator and that no one is eavesdropping on your conversation, Cryptully uses a secret question and answer method.

To authenticate a chat session:

1. When chatting, select "Authenticate Chat" from the options menu.



2. Enter a question and answer that only you are your buddy knows the answer to. Note that the answer is case sensitive.

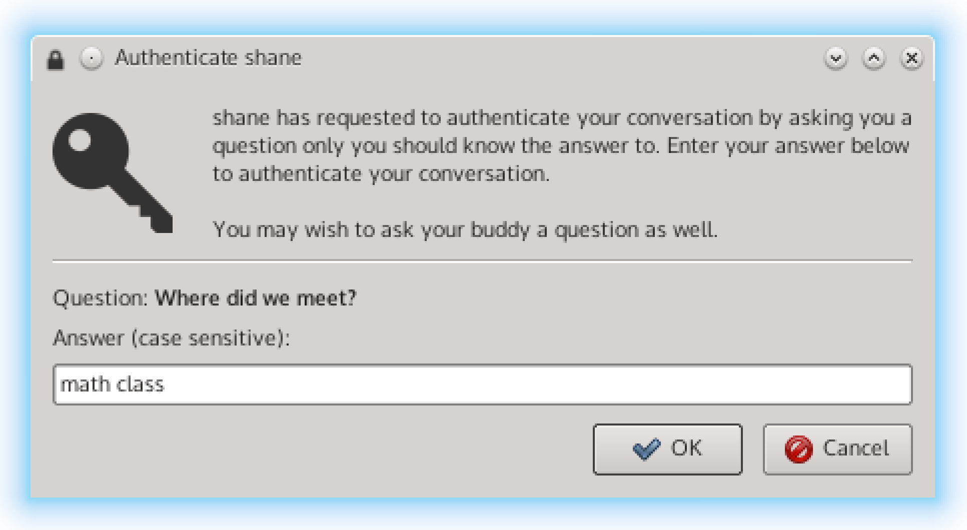


In order to ensure that no one is listening in on your conversation it's best to verify the identity of your buddy by entering a question that only your buddy knows the answer to.

Question: Where did we meet?

Answer (case sensitive): math class

3. Your buddy will then have a window open that allows the answer to the given question to be entered.



shane has requested to authenticate your conversation by asking you a question only you should know the answer to. Enter your answer below to authenticate your conversation.

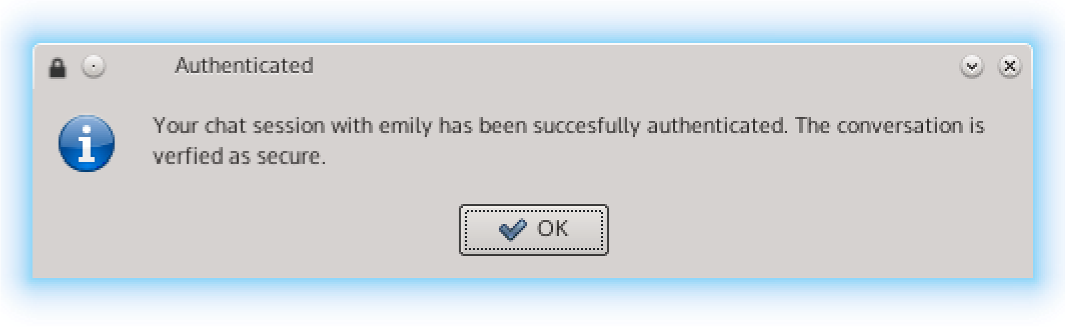
You may wish to ask your buddy a question as well.

Question: **Where did we meet?**

Answer (case sensitive): math class

4. If your buddy entered the same answer as you, the chat will be successfully authenticated. A failure to authen-

ticate means either the wrong answer was entered or someone is listening in on your conversation.



Authenticated

Your chat session with emily has been succesfully authenticated. The conversation is verfied as secure.

5.2.4 Command Line Options (advanced)

Advanced users may utilize command line options of Cryptully:

usage: cryptully.py [-h] [-k [NICK]] [-r [TURN]] [-p [PORT]] [-s] [-n]

optional arguments:

-h, --help	show this help message and exit
-k [NICK], –nick [NICK]	Nickname to use.
-r [TURN], –relay [TURN]	The relay server to use.
-p [PORT], –port [PORT]	Port to connect listen on (server) or connect to (client).
-s, --server	Run as TURN server for other clients.
-n, --ncurses	Use the NCurses UI.

5.2.5 Running Your Own Server (advanced)

If you don’t want to use the default relay server, you can host your own.

This is as easy as downloading a pre-built binary, or getting the source and running Cryptully with the `--server` command line argument.

5.3 Building Cryptully

5.3.1 Get the source

You can get the source from GitHub at https://github.com/shanet/Cryptully.

5.3.2 Dependencies

So you want to build Crpytully from source? Sweet! There’s a few dependencies you’ll need to install first though. The core dependencies are:

- Python 2.7
- PyQt4

- M2Crypto
- PyInstaller 2 (only if you want to build a binary, otherwise it is not needed)

Depending what platform you're building on, you'll need a few more things, but those are documented in the relevant sections below.

5.3.3 Sanity Check

You can do a basic sanity check by doing the following:

```
$ python
>>> import PyQt4.QtGui
>>> import M2Crypto
>>> import curses
```

If you got any errors, something isn't right.

5.3.4 Running Directly

Cryptully is, after all, a Python script so you can run it without packaging it into anything fancy. Once the sanity check passes, just do:

```
$ python cryptully/cryptully.py
```

You can also check out all the command line options with:

```
$ python cryptully/cryptully.py --help
```

5.3.5 Linux

These instructions are for Debian/Ubuntu. They should work on other distros, but the package names will most likely be different.

1. `$ apt-get install python-dev python-qt4-dev python-m2crypto python-stdeb`
2. Download and extract PyInstaller from http://www.pyinstaller.org.
3. `$ cd /path/to/cryptully/`
4. `$ python make.py dist /path/to/pyinstaller/`

If everything went as intended, the packaged application should be in `dist/`.

5.3.6 Windows

1. Download and install Python 2.7 from http://www.python.org/download/releases/2.7.5. You'll probably want to add it to your PATH as well.
2. Download and install of PyWin32 from http://sourceforge.net/projects/pywin32/files/pywin32 (these instructions were tested with build 218)
3. Download and install M2Crypto http://chandlerproject.org/Projects/MeTooCrypto#Downloads
4. Download and install PyQt4 from http://www.riverbankcomputing.com/software/pyqt/download
5. Download and install the Visual C++ 2010 Redistributable Package from https://www.microsoft.com/en-us/download/details.aspx?id=14632

6. Download and install Curses from http://www.lfd.uci.edu/~gohlke/pythonlibs/#curses
7. Download and extract PyInstaller from http://www.pyinstaller.org

It's probably a good idea to do the sanity check as described in the sanity check above at this point.

8. `> cd \path\to\cryptully\`
9. `> python make.py dist \path\to\pyinstaller\`

If everything went as intended, the packaged application should be in `dist\`.

5.3.7 OS X

1. Install Homebrew from http://mxcl.github.io/homebrew (it's the easiest way to get the dependencies)
2. Run `$ brew doctor` to make sure everything is okay. You'll probably need to install the OS X Command Line Tools.
3. `$ brew install python` (OS X comes with a version of Python, but it's best to use the homebrew version)
4. If not done already, set your Python path with: `export PYTHONPATH=/usr/local/lib/python2.7/site-packages`
5. `$ brew install pyqt`
6. Download the relevant M2Crpyto .egg for your version of OS X from http://chandlerproject.org/Projects/MeTooCrypto#Downloads
7. Copy the M2Crypto .egg to `/usr/local/lib/python2.7/site-packages`
8. Download and extract PyInstaller from http://www.pyinstaller.org **At the time of this writing the development version of PyInstaller must be used!** Stable version 2.0 will not work. Future stable versions may or may not.

It's probably a good idea to do the sanity check as described in the sanity check above at this point.

9. `$ cd /path/to/cryptully/`
10. `$ python make.py dist /path/to/pyinstaller/`

If everything went as intended, the packaged application should be in `dist\`.

5.3.8 Unit Tests

Install the Python Mock package first.

Units tests are located in the `src/tests` directory. Running them is as simple as `python make.py test`.

5.3.9 Documentation

The documentation you are reading right now was generated by Sphinx and its source is located in the `docs` folder. To build it into a pretty HTML page just run the following from the `docs` folder

Linux/OS X:

```
make html
```

Windows:

```
.\make.bat html
```

5.4 Protocol

This is an overview of Cryptully‘s protocol for easier understanding the code, or so someone could implement another type of client.

Note: This protocol is highly likely to change in the future.

5.4.1 Basic Properties

- All traffic over the network is formatted as JSON messages with the following properties:
  - `serverCommand`: The command given to the server
  - `clientCommand`: The command given to the destination client
  - `sourceNick`: The nickname of the sender
  - `destNick`: The nickname of the receiver
  - `payload`: The content of the message. If an encrypted message, it is base64 encoded
  - `hmac`: The HMAC as calculated by the sender to be verified against by the receiver
  - `error`: The error code, if applicable
  - `num`: The message number, starting from 0 and monotonically increasing with sequential numbers.
- All commands *are* case sensitive
- After the initial handshake is complete, the connection is kept alive indefinitely in a message loop until either the client or server sends the `END` command.
- The client or server may send the `END` command at any time.

5.4.2 List of All Commands

Server Commands

- `VERSION`: Tell the server what protocol version the client is using
- `REG`: Register a nickname with the server
- `REL`: Relay a message to the client as specified in the `destClient` field

Client Commands

- `HELO`: The first command denotes the initation of a new connection with a client
- `REDY`: The client is ready to initiate a handshake
- `REJ`: If the client rejected a connection from another client
- `PUB_KEY [arg]`
- `SMP0 [arg]`: The question the SMP initiator is asking
- `SMP1 [arg]`
- `SMP2 [arg]`
- `SMP3 [arg]`

- `SMP4 [arg]`
- `MSG [arg]`: The user has sent a chat message
- `TYPING [arg]`: The user is currently typing or has stopped typing
- `END`
- `ERR`

Typing Status

A client may optional give the typing status of the user to the remote client by issuing the `TYPING` command. The `TYPING` command takes one of three possible arguments:

- `0`: The user is currently typing
- `1`: The user has stopped typing and deleted all text from the buffer
- `2`: The user has stopped typing, but left some text in the buffer

5.4.3 Encryption Details

- 4096-bit prime is used to generate and exchange a shared secret via Diffie-Hellman.
- An AES key is the first 32 bytes of the SHA512 digest of the Diffie-Hellman secret. The IV last 32 bytes of this hash.
- All AES operations are with a 256-bit key in CBC mode.
- HMAC's are the SHA256 digest of the AES key and the encrypted message payload. The receiver calculates and verifies the HMAC before attempting to decrypt the message payload.

Socialist Millionaire Protocol

The Socialist Millionaire Protocol (SMP) is a method for determining whether two clients share the same secret, but without exchanging the secret itself. In Cryptully's case, it is used to determine whether a MITM attack has occurred or is occurring and compromised the Diffie-Hellman key exchange protocol.

The innards of the SMP is relatively complex so it is best to defer to the documentation of it's implementation as defined in the Off-The-Record (OTR) protocol version 3.

Cryptully's implementation uses the following commands:

Client A	direction	Client B
	<-	SMP0
	<-	SMP1
SMP2	->	
	<-	SMP3
SMP4	->	

`SMP0` contains the question the initiator is asking. The remaining commands may be sent at any time, as long as they are completed in order and another SMP request is not started before the previous one is completed.

5.4.4 Handshake Details

The commands in the handshake must be performed in the following order:

Client A	direction	Client B
	<-	HELO
REDY	->	
	<-	PUB_KEY
PUB_KEY	->	
(switch to AES encryption)		

The client may reject a connection with the `REJ` command instead of sending the `REDY` command.

5.4.5 Message Loop Details

Clients may send messages any order including multiple messages in a row.

Client A	direction	Client B
MSG	<->	MSG
TYPING	<->	TYPING
END	<->	END

CHAPTER 6

Contributing

For non-programmers:

Even if you're not writing code, you can still help! Submitting any issues or problems you run into at https://github.com/shanet/Cryptully/issues or by emailing shane@shanetully.com. Even reporting something like a section in the documentation not being as clear as it could be or just a typo is helpful.

For programmers:

If you would like contribute to Cryptully, see the *Downloads* page on how to set up a build environment and get the source code. Please any issues encountered at https://github.com/shanet/Cryptully/issues.