P2P Project - Onion

Initial approach report

1. General information

- Team name: Group 61
- Team members:
  - Jan-Cedric Anslinger (ga58jih)
  - Jörn von Henning (ga63vag)
- Project Choice: Onion

2. Choice of Programming Language

For this project, we would like to work with Go (golang). There are several reasons for our choice. First of all, we both love to work with go, and we are both familiar with it, so there is no need for somebody to dive into a new language. Secondly, go is blazing fast, strongly typed, object-oriented, static and supports the developer in creating a lot in a short period. There are already a lot of packages available, so we don't have to start from zero and go code can run on all major operating systems. So from our point of view, it has ideal conditions to work on such a project as ours.

Concerning the operating system, we will work on Mac OSX. Both of us use a MacBook on a daily basis, are familiar with this system and love it. And in stark contrast to windows, we've almost the same freedom as on ubuntu or comparable systems, so there is no need to switch.

3. Build System

We are using the implemented build system of Go.

4. Guaranty of software quality

In the standard library of go, there is already a package called testing. This package is designed for testing, especially benchmarking, and can be easily included in our project. So we will start probably with simple unit tests and table tests at the beginning to check whether everything works as expected. Another benefit of using the testing package of go is that benchmarking can be included easily. So after verifying that everything works as expected, we will benchmark everything to see where are still opportunities to enhance the performance of our program. Moreover, there is a "go standard" which describes how to name functions, attributes etc. so that there is already a collective sense of writing code. Besides, we define our own standard at the beginning and will continuously update the documentation of our project. Last but not least, we try to follow the promoted work routine concerning git to reduce merge conflicts so that our master is always clean and runnable.

5. Libraries

- Net: Net provides a portable interface for network I/O, including TCP/IP, UDP, domain name resolution, and Unix domain sockets.
- Testing: Testing provides support for automated testing of Go packages.
- Crypto: Crypto collects common cryptographic constants.

- Encoding: Encoding defines interfaces shared by other packages that convert data to and from byte-level and textual representations.
- OS: OS provides a platform-independent interface to operating system functionality.

6. License

We intend to use the **MIT License**. We do not persue using the code in any commercial way, hence this project will be published as an Open-Source project. Since the application should be annonymous and unobservable, we figured out that this should be handled with the code likewisely.

7. Programming Experience

- Jan-Cedric Anslinger
    - Development of several production ready APIs
    - Development during iLab2
- Jörn von Henning
    - Development of various APIs
    - Security Knowledge (Android Security, online courses)

8. Share of Workload

- Create and read communication
- Find right amount of peers for a tunnel (check via benchmarking)
- Determine Fixed package size (check via benchmarking)
- function build Packages according to fixed Package size
- Disconnect peers which are sending packages violating the size
- Shutdown and rebuild tunnels
- Switch seamlessly between tunnels
- Init Tunnel: start constructing a tunnel with at least two intermediates
    - function to build connection between peers
    - generate session key (based on both hostkeys)
    - encrypt communication
    - randomly choosing a peer (using RPS)
- Messages:
    - Onion Tunnel Build
    - Onion Tunnel Ready
    - Onion Tunnel Incoming
    - Onion Tunnel Destroy
    - Onion Tunnel Data
    - Onion Error

As seen above, we tried to divide the whole system into several "major tasks". We plan to write the basic function "init tunnel" without all details together because we think that it is crucial that both of us understand 100% this part of the system. Afterwards, we would split the other requirements of this system between us so that we keep on improving the product. This workflow should enhance our understanding and should be easy to implement because we have a solid foundation which gets advanced by both of us. Moreover, we should always have a testable system where small components can be easily changed.

9. Issues

- Shorter registration period, so that one has more for the main project.