**Institut für Informationssysteme**
Abteilung für Verteilte Systeme
VU Advanced Internet Computing (184.269)

**Project Topic 3 – Cloud-based Onion Routing**

# Introduction

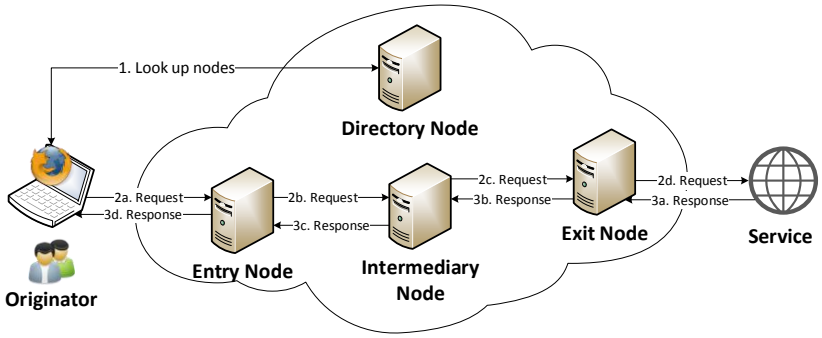


Figure 1: Cloud-based Onion Routing Sketch

Your task in this project is to develop a Cloud-based anonymity solution which implements some basic functionalities of onion routing [1, 2]. Onion routing is a well-known basic principle for providing anonymous communication and has been implemented, e.g., in Tor[1]. The basic idea of onion routing is to obfuscate the identity of a user by routing a message (e.g., a request for a Website) via different intermediary nodes (routers) and decrypting or encrypting, respectively, the message at every single node.

Of course, it is not possible to implement a fully Cloud-based onion routing solution in the lab. Hence, we will focus on particular functionalities:

- Implementation of one central directory node (running in the Cloud – see Figure 1). This directory node also controls the other nodes, i.e., instantiates new chain nodes running in virtual machines (VMs).
- The originator (i.e., the end user's machine) receives a chain from the directory node. This chain will be used to route requests or responses within this onion routing infrastructure.
  - A chain is made up of three chain nodes running on VMs in the Cloud. In Figure 1, the chain is made up of the three nodes "Entry Node", "Intermediary Node", and "Exit Node".
  - The major functionality of these chain nodes is to route requests respectively responses based on routing information received from the originator.
  - For *Requests*: The originator encrypts the message it wants to send. The chain nodes' public keys are used to encrypt the message (three times and in reverse order). The chain nodes decrypt the message and forward it.
  - For *Responses*: The chain nodes encrypt the messages using the public key of the originator, i.e., again encryption is carried out three times. The originator decrypts the message.

[1] `http://www.torproject.org/`

- You will have to provide a user interface at the originator showing how a request is routed across the Cloud.

Also, the result will be way more centralized than onion routing is usually implemented. Simply installing Tor (or a similar solution) on VMs is *not* sufficient – you will have to implement the abovementioned functionalities by yourself.

# Outcomes:

The expected outcomes of this project are three-fold: (1) the actual project solution, (2) a brief paper that analyses the scientific state of the art in anonymity systems, and (3) two presentations of these results (paper and tool).

## Project Solution

The project should be hosted on a public Git repository, whereas the URL to the repository has to be submitted. We recommend either Github[2] or Bitbucket[3]. Every member of your team should have a separate Github or Bitbucket account. It is required to provide an easy-to-follow README that details how to deploy, start and test the solution.

For the submission (see below), you also have to create a virtual machine (Linux or Windows as a vmdk file), which hosts your implemented solution preconfigured with some test data and a short README how to access your implementation. The virtual machines are collected during the final meeting.

## Paper

The paper should provide an overview about the scientific state of the art in anonymity networks. The paper does not have to be restricted to onion routing, if there are promising alternative solutions which could replace or complement the onion routing approach. Note that the paper is not the documentation of your tool – it should mainly discuss scientific papers related to these topics in the style of a seminar paper. However, put more focus on these approaches that are actually relevant / related to your solution and Cloud Computing.

Good starting points for finding related scientific papers are the works cited in this text, Google Scholar[4], IEEEXplore[5] or the ACM Digital Library[6]. Use the ACM "tight" conference style[7] (two columns), and keep the paper brief (4 pages). Stick to bibliography layout demanded as part of the style; not providing the necessary information will lead to a deduction of points.

You do not necessarily need to install a LaTeX environment for this – you can use writeLaTeX[8], a collaborative paper writing tool, as well. However, usage of LaTeX is strongly recommended.

## Presentations

There are two presentations. The first presentation is during the mid-term meetings, and should cover at least the tasks of Stage 1 (see below), the second presentation is during the final meetings and contains all your results. The actual dates have been announced in TISS.

Every member of your team is required to present in either the first or the second presentation. Each presentation needs to consist of a regular e.g., Powerpoint part and a demo of your tool. Carefully think about how you are actually going to demonstrate your tool, as this will be part of the grading. You have 20 minutes per presentation (strict). We recommend to use only a small

[2] https://github.com
[3] https://bitbucket.org
[4] http://scholar.google.at/
[5] http://ieeexplore.ieee.org/Xplore/home.jsp
[6] http://dl.acm.org/
[7] http://www.acm.org/sigs/publications/proceedings-templates
[8] https://www.writelatex.com

part of the presentation for the Powerpoint part (e.g., 5 minutes) and to clearly focus on the presentation of your results.

### Grading

A maximum of 50 points are awarded in total for the project. Of this, up to 25 points are awarded for the tool, up to 10 points are awarded for the paper, and up to 15 points are awarded for the presentations. Grading will be based on the quality and creativity of solutions, presentations, and the paper.

A strict policy is applied regarding plagiarism – both for the paper and the source code. Plagiarism in the paper will lead to 0 points for this part and therefore necessarily decrease the grade for the whole group. Plagiarism in the source code will lead to 0 points for the particular student who has implemented this part of the code. If more than one group member plagiarizes, this may lead to further penalties, i.e., 0 points for the tool.

### Deadline

The hard deadline for the project is **January 25th, 2015**. Please submit a link to your solution Git repository, deployment instructions, the paper, and the presentations as a single ZIP file via TUWEL.

The submission system will close at 18:00 sharp. Late submissions will not be accepted. The only exception is the virtual machine, which is collected during the final meeting.

### Test Cloud Infrastructure

This year, we have access to a grant for Amazon Web Services[9] (AWS), which you can use to deploy and test your solution. Send an email to `aic14@dsg.tuwien.ac.at` if you wish to get access to the AWS cloud environment.

## Stage 1

In the first stage, the focus should be on establishing the routing of a message from the originator to the receiver and setting up the basic infrastructure, including the directory node and the encryption/decryption functionalities. For now, you may instantiate the chain nodes manually and report the chain nodes to the directory node. Use the Amazon AWS SDK[10] to create and manage the VMs.

### Tasks:

1. Familiarize yourself with the Amazon EC2 user console, which can be used to start and terminate new VM instances and to create snapshots, which can be used as images for new VM instances, like nodes.

2. As a next step begin with building the chain node routing functionalities. You need to draw a concept how a chain is built and how nodes will know about how and where to route a message. Be aware that you will need a new chain for *every* message to be sent by an originator!

3. Every chain node has to generate a new public/private key pair at start up. We recommend the Bouncycastle Crypto API [11] for the key generation, encryption and decryption.

[9] `http://aws.amazon.com`
[10] `https://aws.amazon.com/tools/`
[11] `https://www.bouncycastle.org`

4. Implement the basic functionalities of the directory node, i.e., the actual directory functionalities: Which chain nodes are available? Where are these nodes located? The directory node needs to be able to answer corresponding requests from a user's machines, i.e., to provide three chain nodes if requested.

5. To forward a *request* message, each chain node needs to decrypt the received message, to obtain the location of the next directory node. To forward a *response* message, each chain node needs to encrypt the received message.

6. Both chain nodes and the directory node have to be deployed as EC2 instances.

7. Provide a basic Restful or SOAP service which provides random quotes from a given text file.[12] This service should also run on an EC2 instance.

8. Provide a basic originator software, i.e., a software which is able to request information from the service via a chain. For Stage 1 it is sufficient to only implement a command line interface.

9. Provide logging functionalities for the chain nodes; you should be able to show which messages have been routed through/by a chain node.

## Stage 2

In the second stage, the directory node will be augmented by more sophisticated functionalities. Most importantly, the directory node will be able to control and manage the Cloud-based onion routing network.

### Tasks:

1. Extend the directory node: The directory node should be able to control the chain nodes, i.e., automatically instantiate exactly 6 chain nodes which are used for routing. The directory node monitors these chain nodes (a simple ping-based approach is sufficient for this). If one chain node is no longer available, the directory node will re-instantiate VMs until there are 6 chain nodes again.

2. Also, in case of the failure of a VM, "lost" request messages have to be resend.

3. The directory node should provide basic load balancing: If an originator asks for "new" nodes to build a chain, the directory node will choose 3 chain nodes based on some rules you have to define.

4. Provide a simple, Web-based UI which allows the user to request the abovementioned service. This UI needs to be able to show the response (as plain text), and also how the messages were routed along the chain and where the chain nodes are located.

## References

[1] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining Light in Dark Places: Understanding the Tor Network. In *8th International Symposium on Privacy Enhancing Technologies*, pages 63–76, 2008.

[2] Jian Ren and Jie Wu. Survey on anonymous communications in computer networks. *Computer Communications*, 33:420–431, 2010.

[12] E.g., you may choose a file from `http://www.textfiles.com/science/`.