Computer Science Tripos – Part II – Project Proposal

An implementation and evaluation of Loopix, an anonymous communication system

Jun Siang Cheah, Christ's College

Originator: Dr Alastair Beresford

19 October 2017

Project Supervisor: Dr Alastair Beresford

Director of Studies: Dr Richard Mortier

Project Overseers: Dr Timothy Jones & Prof Marcelo Fiore

Introduction

Anonymous communication systems such as Tor are becoming increasingly important in the current age of pervasive data collection, surveillance and censorship, allowing for privacy and anonymity when parties are trying to collect as much data as possible on individuals, either for commercial exploitation or government surveillance.

However, the most widely used anonymous communication system, Tor, is vulnerable to attacks such as traffic correlation by a global passive adversary and corrupt nodes performing active attacks to deanonymise users. Government agencies such as the National Security Agency (NSA) and Government Communications Headquarters (GCHQ) have already demonstrated the ability to deanonymise a small fraction of Tor users.[1] Alternative systems that are not vulnerable to a global passive adversary tend to be high-latency, low-bandwidth, which severely limits possible applications.

Loopix[2] is a newly proposed protocol that provides medium-latency, low-bandwidth communication that provides strong anonymity against both active attackers and global passive adversaries. The medium-latency property of the system allows for both low-latency applications such as instant messaging and high-latency applications such as email. Another feature of Loopix is the ability to store messages for offline clients. This is helpful for peer-to-peer communications when it is difficult to have both clients online at the same time such as mobile devices.

Currently, the reference proof of concept implementation is written in Python with a dependency on Sphinx[3]. A Python implementation works well enough for server nodes where there are not many constraints, however, the Python code is not portable for mobile devices. As such, I plan to implement Loopix as a Java library, which can then run on any system with a JVM installed such as Android. After the implementation is complete, I would evaluate the performance of the system (bandwidth, latency, and complexity) and validate some of the privacy claims in the original paper.

Starting point

Currently, there exists a Python implementation of Loopix which I plan to use to test my own implementation against and would allow for a faster iteration cycle as I wouldn't need a complete implementation to begin testing. I will also refer to the existing implementation for details that are specified in the paper. I will also rely on the BouncyCastle Java library for cryptographic primitives.

The cryptography used will relate to the Security I and II courses, and I will draw upon the Part IB Mathematical Methods course for the statistical theory used in Loopix. I have some experience with the theory behind mix networks from working with Tor.

I have written an existing Android application that currently uses Tor for peer-to-peer communication which can be adapted to use Loopix instead for demonstration purposes.

Project structure

The aim of this project is to provide a Java library implementation of Loopix, therefore taking advantage of the ubiquity of JVMs to run Loopix on a larger range of devices, with a primary goal of allowing Android applications to use Loopix for communication.

The project can be broken down into the following sub-projects:

1. **Implementation of the Sphinx library:** Loopix uses the Sphinx mix format as the interchange format for messages, thus a Java Sphinx library needs to be written before any work on Loopix can proceed. Sphinx uses AES, Diffie-Hellman and SHA256 cryptographic primitives, which are implemented by the BouncyCastle library. I will use the existing implementation to test my own implementation by generating and parsing messages on both sides.
2. **Implementation of the Loopix library:** This library will the at the very minimum provide a client implementation of Loopix that is compatible with the existing Python implementation for servers. The Loopix architecture calls for 3 components: the client, provider and mix nodes. All 3 share the same message processing pipeline and only differ in behaviour upon parsing the decrypted message.
3. **Demonstration application:** An application that uses the resulting Java library to demonstrate that the library is working. A possible application to implement is an instant messaging application.
4. **Evaluation:** The final part of the project is to benchmark the performance of the implementation in terms of bandwidth and latency overheads. I will also need to verify some of the claims made in the paper.

Success criteria

The project will be a success if:

- I have completed a Java implementation of the Sphinx library.

- I have completed at the very minimum a client implementation of Loopix in Java.
- I have created a demo app that uses the library to send messages.
- I have produced an evaluation of my Loopix implementation, in terms of bandwidth throughput, bandwidth overhead, latency overhead, complexity overhead and privacy metrics.

Possible extensions

If I achieve my main result early, then possible extensions include:

- **Discoverability:** At the moment, all nodes in the network must be preloaded with a list of all the other nodes in the network, including their IP address and public keys. This simply does not scale and fails to deal with failures with some of the nodes in the network and mobility of real clients. A potential solution could be to borrow Tor's directory authority model and implement it within Loopix. Implementing a directory also allows for a scheme similar to Tor's .onion domain names to refer to other nodes, rather than their IP address.
- **Provider resource starvation:** Currently, providers store all messages in memory with no eviction strategy. This is bad in many ways, as providers can run out of memory due to either a malicious actor flooding the provider with messages, an organic increase in traffic or users never coming online to clear their inbox. I could research into methods for mitigating and handling this issue, as a user would find messages disappearing a bad experience.
- **Android application demo:** I could build a demo app on Android which is where Loopix shines with its offline message storage.

Timetable and milestones

Weeks 1-2 — 16 Oct - 29 Oct — Michaelmas term

- Familiarise myself with the Sphinx and Loopix papers and implementation.
- Setup test scripts for comparing outputs from Sphinx/Loopix.

Weeks 3-4 — 30 Oct - 12 Nov — Michaelmas term

- Implement the Sphinx library in Java.

Milestone 1: Sphinx library completed

Weeks 5-8 — 13 Nov - 17 Dec — Michaelmas term + Christmas vacation

- Implement the client portion of the Loopix library in Java.
- Create a command line wrapper around the library to test with.
- Start initial testing using the existing Python implementation to setup providers and mix nodes on a small scale.

Milestone 2: Client portion of the library done, and I can send and receive messages using the library.

Weeks 9-10 — 18 Dec - 31 Dec — Christmas vacation

- Implement the provider/mix node portion of the Loopix library.

Weeks 11-13 — 1 Jan - 21 Jan — Christmas vacation + Lent term

Contingency time set aside for any major issues found up to this point.

Weeks 14-15 — 22 Jan - 4 Feb — Lent term

- Implement a demo application that uses the Loopix library.
- Submit Progress Report

Milestone 3: A working demo that can be shown to my supervisor/DoS.

Weeks 16-17 — 5 Feb - 18 Feb — Lent term

- Research existing attacks on anonymous communication systems.

Weeks 18-20 — 19 Feb - 11 Mar — Lent term

- Evaluate the project, recording any necessary data.

Milestone 4: Evaluation data gathered

Weeks 21-25 — 12 Mar - 15 Apr — Lent term + Easter vacation

- Write and submit my draft dissertation.

Milestone 5: Draft dissertation complete!

Resources required

A list of required resources:

- My own machines:
  - Desktop computer (AMD Ryzen 5 1600, 32GB RAM, Windows/Ubuntu 16.04)
  - Laptop (Intel i7-4720HQ, 8GB RAM, Windows)
  - Backup desktop (Intel i5-6500, 16GB RAM, Windows/Ubuntu 16.04)
- My own dedicated servers (2x Intel i5-2300, 16GB RAM, Ubuntu 16.04)
- BouncyCastle library[1]
- Loopix source[2]

Backups will be provided by Dropbox and OneDrive. Revision control will be provided using GitHub, with a backup remote on Bitbucket. I can switch to using the MCS

[1]https://www.bouncycastle.org/
[2]https://github.com/UCL-InfoSec/loopix

machines if my personal machines fail, and use student credit to run VMs with cloud providers if my servers fail.

References

[1] National Security Agency. Tor Stinks. `https://edwardsnowden.com/wp-content/uploads/2013/10/tor-stinks-presentation.pdf`, 2012. [Online; accessed 1-October-2017].

[2] Ania Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. *arXiv preprint arXiv:1703.00536*, 2017.

[3] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 269–282. IEEE, 2009.