Collective Signing for Next Generation Dissent

William Edward Bailey, III

January 17, 2015

Faculty Advisor: Bryan Ford
Dept. of Computer Science

Background

In a world of increasing interconnectedness via the Internet, the need for systems to provide better anonymity and privacy is more than ever before. Numerous organizations are developing sophisticated tools for tracking and identifying web surfers. Whether they be advertising companies profiling users or totalitarian governments suppressing freedom of speech, modern users are seeing their privacy evaporate at an alarming pace. To counteract this trend, more effective technologies are needed to ensure that those who wish to remain anonymous can successfully maintain their online privacy.

Overview

I will help implement collective signing for the next generation of Dissent. Currently headed by my advisor Bryan Ford, Dissent is “an accountable anonymous group communication system [...] whose goal is to offer...] strong, provable security guarantees with reasonable efficiency”. A major issue with current solutions to anonymity is that they are vulnerable to powerful adversaries (such as government controlled ISPs) who can perform traffic analysis and other sophisticated attacks. To overcome this, Dissent uses a cryptographic system based on the
dining cryptographers (DC-net) algorithm, which uses shared secrets between clients to preserve anonymity as long as at least one other client is honest. The current version of Dissent successfully prevents traffic analysis attacks at the cost of high latencies.

The next generation of Dissent aims to reduce latency to support widespread use on the order of millions of users. Dissent currently uses a server-client system in which clients share DC-net secrets with only servers to reduce network traffic. In order for this system to work in a dynamic environment, clients will need a list of all Dissent servers they can use to preserve their privacy. Dissent must ensure that this list is as accurate as possible and that a malicious server can not exclude others from the list. Hence, the need for collective signing. The server list will be signed by a key generated from shared secrets of all current servers. Since all servers contribute to signing and approving this list, a subset of dishonest servers can not manipulate it.

To improve the fault-tolerance of Dissent, it is important for the protocol to make progress in the absence of some servers. In the current version of Dissent, the entire protocol halts if a single server goes offline. To solve this problem, I will develop a “life insurance protocol”. Before a server can perform work, it must distribute shards of its shared secret to a certain number of other servers that will act as insurers. In the event that the server goes offline, a client can complain to the insurers, whose shards can be recombined to form the original secret. Hence, another server can continue in place of the original. This robustness will be vital in a world where servers may join or leave the network, experience unforeseen failures, or be taken down for maintenance.

I will contribute to collective signing by creating this insurance policy. If time permits, I may also work on other portions of collective signing.
Specification

I will program a Go library for the life insurance protocol. It will consist of:

- A function for creating shards of a shared secret and sending them to insurers
  - Parameters
    - A shared secret using Verifiable Shamir Secret Sharing
    - A list of possible nodes to server as insurers
    - A method for selecting the insurers
    - A verification method for ensuring that the choice of insurers is acceptable
    - A value to denote the total number of insurers needed
    - A value to denote the number of shares needed to reconstruct the secret
- A function that produces a list of receipts from insurers certifying a server has taken out a policy with them
- A function for clients to request that a server perform some work for them
  - Parameters
    - The type of work to accomplish
    - A time-limit
- A function to contact the insurers if the time-limit for a server to complete some work expires (in which case the server may be dead)
- A function for the insurers to verify if the server is dead.
  - If the server is not dead, it forwards the finished work from the server to the client.
  - Otherwise, it gives its shared secret from the server to the client.
Deliverables

I will fully implement and extensively test the library above so that it will be ready for use by Dissent. Depending upon how long this takes, I may extend this project by:

- Working on collective signing more generally
- Developing a good function for choosing insurers
- Defining what an acceptable choice of insurer is and developing a function to verify that a choice is acceptable