PriFi: A Low-Latency and Tracking-Resistant Protocol for Local-Area Anonymous Communication

1. Introduction

This repository implements PriFi, an anonymous communication protocol with provable traffic-analysis resistance and small latency suitable for wireless networks. PriFi provides a network access mechanism for protecting members of an organization who access the Internet while on-site (via privacy-preserving WiFi networking) and while off-site (via privacy-preserving virtual private networking or VPN). The small latency cost is achieved by leveraging the client-relay-server topology common in WiFi networks. The main entities of PriFi are: relay, trustee server (or Trustees), and clients. These collaborate to implement a Dining Cryptographer's network (DC-nets) that can anonymize the client upstream traffic. The relay is a WiFi router that can process normal TCP/IP traffic in addition to running our protocol.

For more details about PriFi, please check our WPES 2016 paper.

Warning: This software is experimental and still under development. Do not use it yet for security-critical purposes. Use at your own risk!

Structure

The current code is organized in two main parts :

1) *PriFi-Lib*, which is network-agnostic; it takes an interface "MessageSender" that give it functions like SendToRelay(), SendToTrustee, ... and ReceivedMessage()

This approach helps PriFi development, as without the network, the protocol becomes much simpler (at least 50% less line of codes); the goal is to develop new functionalities without knowing anything about the network layer, or SDA.

2) *PriFi-SDA-Wrapper*, that does the mapping between the tree entities of SDA and our roles (Relay, Trustee, Client), and provides the MessageSender interface discussed above.

This binder now uses SDA, which is convenient, but could use any library. In particular, it *could* use SDA *and* direct TCP/UDP streams in parallel for performance reasons.

Running PriFi as a simulation

Currently, the PriFi code can be run through a simulation. To run it first make sure that:

- `$GOPATH/src/github.com/dedis/cothority` is on branch `prifi` and pulled to the latest version.
- `$GOPATH/src/github.com/dedis/crypto` is on branch `master`

and pulled to the latest version.
- `$GOPATH/src/github.com/lbarman/prifi_dev` is on branch `PriFi-SDA` and pulled to the latest version and that you are in this directory.

We can now run the simulation:

```
./run-prifi-via-simulation.sh
```

It is possible to browse through PriFi using the SOCKS5 integration. To use it launch the SOCKS5 server:

```
go run socks/run-server.go
```

and configure your browser to use a SOCKS5 proxy on `localhost` on port `6789`.

Simulation parameters

The configuration for the simulation is located in the Cothority repository, in the file `simul/runfiles/prifi_simple.toml`. This file contains the following parameters:

- `Servers (int)`: Number of logical cothority entities. There is a trick here, when using udp in local (with channels) we should specify ONE server so all entities are run on the same name space, and share the channels. Otherwise, this parameter does not have much effect.

- `Bf (int)` : Branching factor of cothority's communication tree. Here we want a flat tree (1 layer with relay, all nodes directly below).

- `Rounds (int)` : Numers of round to simulate (not PriFi).
- `CloseWait (int)` : Wait that much (in milliseconds ?) before killing a cothority protocol and going to next simulation round.
- `DataOutputEnbaled (bool)` : Enables the link from and to the socks proxy.
- `NTrustees (int)` : Number of trustees.
- `CellSizeUp (int)` : Size of upstream data sent in one PriFi round (?)
- `CellSizeDown (int)` : Size of upstream data sent in one PriFi round (?)
- `RelayWindowSize (int)` : Number of ciphers from each trustee to buffer
- `RelayUseDummyDataDown (bool)` : When true, the relay always send CellSizeDown bits down. When false, it may send only 1 bit.
- `RelayReportingLimit (int)` : Unused, was for the statistics.
- `UseUDP (bool)` : Enable or disable UDP broadcast for downstream data (?)
- `DoLatencyTests (bool)` : Enable or disable latency tests.
- `ReportingLimit (int)` : PriFi shuts down after this number of rounds if not equal to `-1` .
- `Hosts (int)` : When NOT in localhost (but in DeterLab), number of physical machines to deploy cothority (and PriFi)

on.

Reference configuration file

The following configuration should allow you to run the simulation and browse through the SOCKS5 proxy without problems (you can copy it in `simul/runfiles/prifi_simple.toml`):

```
Simulation = "PriFi"
Servers=5
Bf = 100
Rounds = 10
CloseWait = 6000
DataOutputEnable = true
NTrustees = 2
CellSizeUp = 1000
CellSizeDown = 10000
RelayWindowSize = 10
RelayUseDummyDataDown = false
RelayReportingLimit = -10
UseUDP = false
DoLatencyTests = false
ReportingLimit = 10

Hosts
5
```