PriFi: a traffic-analysis resistant, low-latency anonymous WLAN

Ludovic Barman
EPFL

Advisor: Prof. Bryan Ford
DEDIS, EPFL

Advisor: Prof. Jean-Pierre Hubaux
LCA1, EPFL

Advisor: Italo Dacosta
LCA1, EPFL

In an effort to thwart today's mass surveillance, and to provide better alternatives to entities who value anonymity, we propose a novel anonymous communication network that leverages on a specific (but common) network setup to provide both *traffic-analysis resistance* against a global passive adversary, and *low-latency communication*. We implement our protocol and show that it is practical in the designed situation: it scales well in the number of clients, and anonymizes up to 20 Mbps with a latency of 40ms in a common setup.

1. Introduction

In today's world of mass internet surveillance [1, 3, 16] and control of information [17, 21, 28], an increasing number of people feel unsafe while communicating on the internet [9, 13]. Freedom of speech, a key to modern and open societies, is being threatened by the constant recording of online human interactions by states and other powerful entities. Encryption only solves partially the problem, as it does not hide the entities communicating. The remaining contextual information, the *metadata*, allow the aforementioned entities to efficiently spy and track people on a scale never experienced before; a dramatic situation, especially in oppressive regimes. One of the ways to reclaim freedom of speech is via *anonymous communication networks* that allow people to share information while being indistinguishable from their peers, and hence untraceable by an eavesdropping entity.

The most widespread anonymous network, Tor [27], was simply not designed to protect against global surveillance [30]. It is now well-established that state-level adversaries can and do *de-anonymize* Tor users [20, 26], making this tool unsuitable for sensitive communication in today's world. We argue that there is a tension between low-latency communication network, bandwidth usage, and traffic-analysis resistance; Tor focuses on the first two, leaving the door open for a global eavesdropper to track people.

In our work, we build a protocol based upon Dining Cryptographer's networks, or *DC-nets* [4], anonymous communication networks with *provable anonymity* against a global passive adversary. Those networks have been thoroughly studied [7, 15, 35] due to their perfect anonymization property, but were considered too impractical for everyday use, notably because of their high bandwidth usage.

Hence, we present PriFi[1], an application-agnostic protocol with provable anonymity that creates an anonymous communication network on a (W)LAN[2]. In addition, we show that under certain (reasonable) assumptions, we achieve *low-latency*, making our network suitable for all-purpose communications (similar to a VPN service). By deploying it on a real infrastructure, we show that when tailored to a (W)LAN network, the protocol's high bandwidth usage is not problematic and becomes practical enough for real-life applications; in a typical scenario, we measured anonymous traffic speeds up to 20 Mbps, sufficient for most traffic going out of (W)LANs.

Finally, we present common real-world situations where the deployment of PriFi would not only perform well, but also bring substantial benefits: IoT networks, and banks, company networks.

This paper is organized as follows: First, we explain the fundamentals of DC-nets in Section 2. We describe our system, PriFi, along with the security properties and the threat model in Section 3. Section 4 and 5 describe the deployment setup and the benchmarks, and the results are further discussed in Section 6. Finally, we explore the real-world applications in Section 7, give an overview of the related work in Section 8, and conclude by presenting potential directions for future work in Section 9.

2. Background

2.1. Dining Cryptographer's networks

DC-nets were first introduced by Chaum et al. [4]. In the simplest version, a DC-net is composed of $N > 2$ interconnected users. In the setup phase, every pair of users (u, v) communicate and derive a shared secret $s_{u,v}$ (for example, via Diffie-Hellman). In the communication phase, up to one

[1] *Pr*ivacy-protecting Wi-*Fi*.

[2] We use the notation *(W)LAN* to denote both Wireless and wired LAN networks. The mention W is kept (instead of LAN, simply) due to the extra optimizations possible when using a WLAN network.

user w is allowed to communicate a message m of length equal to the length of the secrets $s_{u,v}$. In order to enable w to communicate his message anonymously, every user u computes the bitwise xor of all the secrets $s_{u,v}$ he has, and broadcast the result S_u to the network; the user w additionally xors b into the sum S_w before broadcasting. One entity is responsible for collecting the N S_u values, and xor'ing them to the final result S, which is sent to the network as the outcome of this communication round. Notice that S is equal to b, since every secret appears exactly twice, and that no information about w is leaked.

To further illustrate the concept, we take the following example: Alice, Bob, and Claire just finished the setup phase of the DC-net, and have already exchanged the secrets s_{AB}, s_{AC}, s_{BC}. We consider those secrets to be fixed-length, random bit-strings. To compute the output, Bob and Claire both xor the two secrets they have, yielding $S_B = s_{AB} \oplus s_{BC}$, and $S_C = s_{AC} \oplus s_{BC}$, respectively. Alice, who wants to communicate the message m, sets $S_A = s_{AB} \oplus s_{AC} \oplus m$. All three users broadcast the sum S_i, and when they received two messages, they compute the final sum $S = S_A \oplus S_B \oplus S_C$, which is equal to m. Notice how Claire is completely unable to tell if Alice or Bob embedded m in S_i, as both S_A and S_B are the sum of at least one pseudo-random value with other values.

3. System Design

3.1. Model

We consider a set of clients, or users $\mathbb{U}$ who want anonymity in their internet communications. They are connected to the internet through a relay, or router $\mathbb{R}$, which processes normal internet traffic in addition to running our protocol.

In addition, we introduce third-parties called *trustees* (presented in Subsection 3.5) whose role is to help the users and the relay.

Users, trustees and the relay (Figure 3.1) will jointly run a protocol called PriFi. From the point of view of the users, it will act as a VPN-service, receiving data from and sending data to the applications running on the user's computer. The relay will act as the other end of the VPN, sending data to the Internet, or to the other users.

3.2. System properties

1. The most important property is **traffic-analysis** resistance, for the reasons explained in the introduction.
2. Furthermore, we want **low-latency**. The time taken by the protocol to anonymize a packet is critical, and should be as low as possible (in the order of milliseconds) to support any kind of traffic on top of our protocol.
3. We want the protocol to be tailored for Wireless **LAN** and wired LAN, supporting between 0 and 50 nodes under normal conditions. The idea is to have an anonymity set organized around one router.
4. Finally, we want the protocol to support the current asymmetry found in the internet usage of most user, and allow **downstream-intensive communication** (while having of course as much performance as possible for the upstream traffic).

3.3. Security goal

Anonymity. We address only the most important security property: the untraceability of the sender. Formally speaking, given two honest users u_0 and u_1 in a set of users $\mathbb{U}$, given the adversary $\mathbb{A}$ defined below (Subsection 3.4), and given a time-slot T, we define the the following game: the environment picks $i \in [0,1]$ with uniform probability; user u_i crafts the message $m_i = 0$, and user u_{1-i} sets m_{1-i} at random in the message space. All parties collectively run Anonymize as specified below, which outputs $m_i \oplus m_{1-i} = m_{1-i}$. Once the round finished, the adversary guesses $\widehat{i}$. Let $p = \Pr[i = \widehat{i}]$ be the probability that $i = \widehat{i}$. The adversary wins if $\epsilon = |p - \frac{1}{2}|$ is non-negligible.

Confidentiality and Authentication. We consider that encryption can always be used on top of the anonymity network, providing authentication and content confidentiality between parties if needed.

3.4. Attacker model

We define a polynomial-time passive attacker $\mathbb{A}$ that 1) controls the network and 2) controls any of the aforementioned entities: clients, servers, or relays. For simplicity, we define the set of entities controlled by the adversary as static, i.e. an entity is either controlled by $\mathbb{A}$, or it will never be. We further define $\mathbb{U}_H \subseteq \mathbb{U}$ the set of honest users, i.e. users not controlled by $\mathbb{A}$. We analyze cases where $\|\mathbb{U}_H\| \geq 2$, since otherwise there is a unique honest client whose communication is trivially identifiable by $\mathbb{A}$. As usual, we do not try to give any guarantees to malicious or corrupt parties.

Denial of service. In this short project, we do not consider *outsider denial of service attacks*, i.e. DoS from entities not described in the system. We will briefly talk about insider accountability in Subsection 3.8.

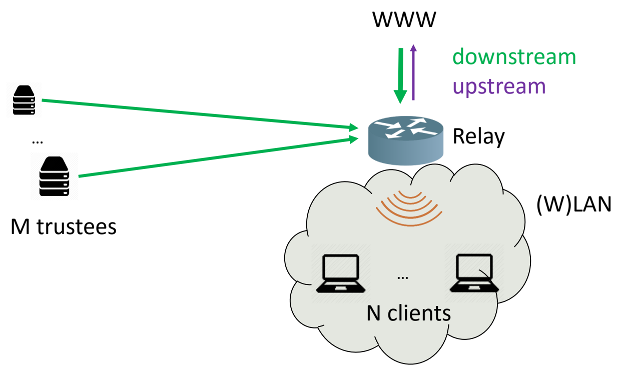


Figure 3.1: System setup

3.5. System description

Let us describe the components of a single PriFi network.

We consider a set of N clients, or users $\mathbb{U}$, a set of M trustees $\mathbb{T}$ whose task is to assist the clients in the anonymization process, and a relay $\mathbb{R}$ who organizes the clients and acts as a gateway to and from the anonymous network. The setup is depicted in Figure 3.1

The trustees are considered with the anytrust model [34]: at least one of them is considered to be honest, and they are all considered to be available at all time.

Let us split the protocol in two parts: We define *downstream* communication as the data from the Internet to one of the clients, and *upstream* communication as the data from one of the clients towards the Internet. The downstream data is simply broadcasted from the relay to every client; no other protocol takes place. The upstream data goes through our anonymization protocol Anonymize, run by the aforementioned entities, and is outputted by the relay towards the Internet.

3.6. Protocols

Setup protocol. During this protocol, the users contact the relay, who broadcasts a list of trustees and their public keys. Clients then agree on the set of trustees and verify the authenticity of the public keys[3]. They announce their choice to the relay, along with their public key. The relay then connects to the specified trustees and announces the list of clients, along with some operational parameters.

Then, clients and trustees exchange or derive the following secrets: each trustee $t \in \mathbb{T}$ and each user $u \in \mathbb{U}$ derives a secret $s_{u,t}$; the secret is then used in combination with a pseudo-random number generator (PRNG) in order to obtain a stream of pseudo-random bits, from which users and trustees will compute their ciphertexts.

Anonymize protocol. The anonymization protocol works in time slots, where clients send the *client ciphers* C_{u_i} to the relay, and so does the trustees with the *trustee ciphers* C_{t_i}. The time-slot ends when the relay collected exactly one cipher from every client and trustee.

Formally speaking, the anonymization protocol is a distributed protocol that outputs one message m per time-slot T, given the users' and the trustees' ciphers $C_u(T)$ and $C_t(T)$ for this time-slot.

$$m(T) \leftarrow \underline{\text{Anonymize}}(C_{u_1}(T), \cdots, C_{t_1}(T), \cdots)$$

Resync protocol. So far, DC-nets have been described using static sets of clients and trustees. While trustees can be in fact considered to be always available (the extension to error-prone trustee is almost similar to the case of error-prone clients), clients can fail, hang, and purposely disconnect; those actions *break the current round*, that is, for this current timeslot T, $\underline{\text{Anonymize}}(\cdots) \rightarrow \emptyset$. It can be easily seen that there is no easy fix to recover the lost data $m(T)$ for this round, as it is xor'ed with a pseudo-random secret that is *not* canceled out in the final xor.

When this happens, client needs to redo the Setup protocol where all the clients are discovered, and the keys are exchanged. During this time, the DC-net cannot process upstream data.

Schedule protocol. Once the DC-net is ready, and clients are ready to communicate, one last question must be answered: which client gets to transmit a message during which time slot. Perhaps the simplest way would be to allow clients to try, and retransmit if a collision is detected (i.e. if $m(T)$ is not the client's message); however, this re-introduces unneeded collision detection (on top of the network protocol used).

A more efficient way is to *schedule* which client gets which time-slot. It is critical that this mapping is *secret*, as otherwise it completely removes all the anonymity properties from the DC-net. For that purpose, we use an oblivious permutation protocol, where all the trustees help in permuting a set of public keys chosen by the clients, in a way that *no one* knows the full final permutation if at least one of the trustees is honest. We use the Neff-Shuffle [24], adapted to elliptic curves, which produces oblivious verifiable shuffle. The output is a set of different public keys; clients are only able to recognize their own public key in the set, other keys look unrelated to all parties without the associated private key.

[3]The distribution of identities and public keys, and the prevention of a man-in-the-middle attack at this level, is out of scope of this short project. We assume all entities received the correct public keys.

This is a very basic version of a scheduling protocol, as it simply assigns one fixed-size slot per client. Several extensions could be created, including client negotiating for a certain number of slots, or variable-length slots, thus distributing the available bandwidth more efficiently.

3.7. Network setup

Our system is optimized for a specific, but common network setup. See Figure 3.1.

1. the set of users $\mathbb{U}$ and the relay $\mathbb{R}$ are all on a (W)LAN, with a latency of at most LOW_LAT (typically in the order of 10ms).
2. the bandwidth HIGH_BW available in this LAN is *relatively high* (typically in the order of 100Mbps for WLAN 802.11*n*)
3. the set of trustees $\mathbb{T}$ and the relay $\mathbb{R}$ are *geographically distant*, with a latency of at most HIGH_LAT (typically in the order of 100ms).
4. the bandwidth from the Internet (hence, from the trustees) to the relay, that is, the downstream bandwidth, is significantly higher than the upstream bandwidth.

Informally speaking, the bandwidth available is *big* 1) in the (W)LAN and 2) downstream, from the Internet to the relay (this includes the traffic from the trustees to the relay).

PriFi fits nicely in this setup. Like other DC-nets, its bandwidth requirement is high, but the only links used by the DC-net are 1) the links from the trustees to the relay, and 2) the links inside the (W)LAN.

Notice how the only tradeoff between downstream speed and upstream speed usually happens because of the link to the ISP (green and purple links), and *not* in the LAN where the bandwidth is much higher. Indeed, there is a tradeoff between the *(anonymized) upstream throughput*, which requires M times this throughput on *downstream* connection from the trustees to the relay. To summarize, to send 1 anonymized bit upstream, the relay downloads M bits from the relay. In practice, this is not a problem, as the bandwidth allocated by the ISP is also asymmetrical - the downstream speed is much higher than the upstream speed.

3.8. Security analysis

Anonymity. PriFi is a special case, with a specific setup, of the general Chaumian DC-net [4] concept. In particular, if we consider both the relay and the trustees to be clients as well, we are exactly in the default DC-net configuration, with the following twists:

- Only some parties have secrets exchanged with some others, but this was already proposed in [4] as a way to reduce the number of keys.
- All "trustees" and the "relay" (which here are just names for Chaumian DC-nets *clients*) do not participate in making the anonymity set bigger, as by design they do not communicate by themselves. However, DC-nets guarantee that even if the anonymity set is as small as two clients, given a correct key and secret setup, they are indistinguishable from each other.

Hence, we argue that we are in a sub-case of the Chaumian DC-net, and that the security properties and proofs from [4] can be adapted to our system in a straightforward manner.

Insider attack & Accountability. Traditionally, DC-nets have been known for poor insider-attack resistance; if a malicious user u using the protocol decides to flip any bit on his secret sum C_{u_i} before sending it to the relay, the corresponding bit in the output $m(t)$ will also be flipped; moreover, by design of the protocol, there is no easy way to trace back this bit to u. Fortunately, the authors of [7] propose a variant of the protocol where honest users can react if the output $m(t)$ is not as expected, and obliviously *blame* the incriminated user, without de-anonymizing the honest users. Note that this protocol does not need to be executed too often: once the insider has been found, he is kicked out of the network; eventually, all malicious users are expelled of the DC-net.

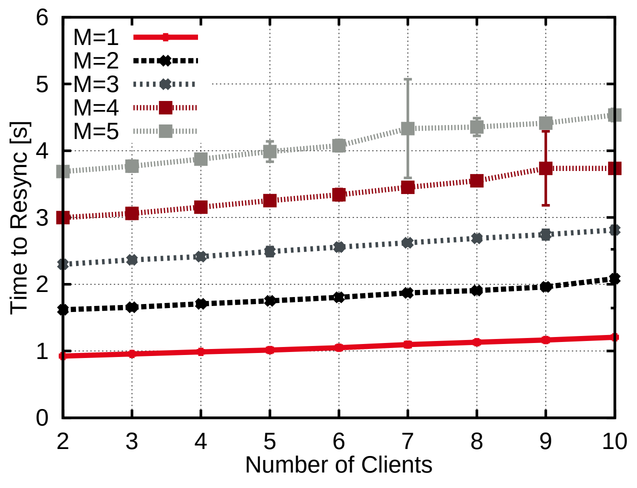
In our situation, this could apply to both malicious users and malicious trustees.

4. Deployment setup

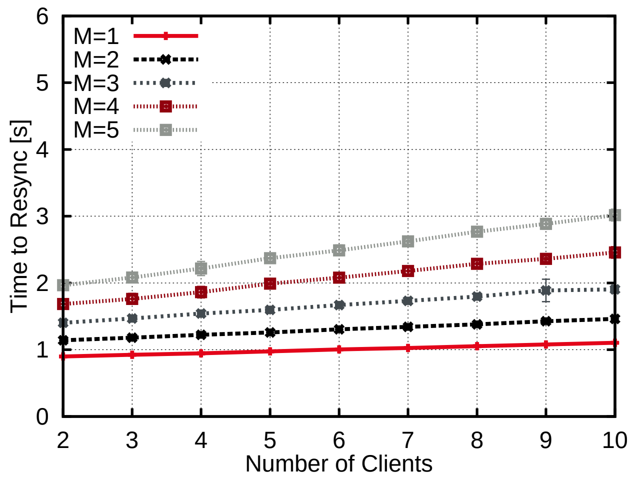
We implemented a prototype of PriFi in Go [19]. We relied on NIST P-256 elliptic curve [25] for asymmetric cryptographic operations, along with AES 128 bits for symmetric cryptography. The hash function used is SHA-256, and we relied on Go's random number generator.

We deployed our protocol to the Deterlab infrastructure [10], with the following configuration: N clients, M trustee servers, and 1 relay. They all run on different machines which have the same specifications: a 2.4 GHz processor Intel Xeon X3 LP, with 16 GB of RAM.

The machines are deployed in two LANs, one for the trustees, with a latency of 100ms, and one for the clients, with a latency of 10ms. The measured latencies are 106 ±0.5 ms and 16 ±1ms (both over 10 samples), respectively. The relay belongs to both LANs. All network links are 100Mb/s Full Duplex.



(a) Naïve version



(b) Optimized version

Figure 4.2: Time to resync, i.e. the time when upstream communication is stopped due to a change in setup.

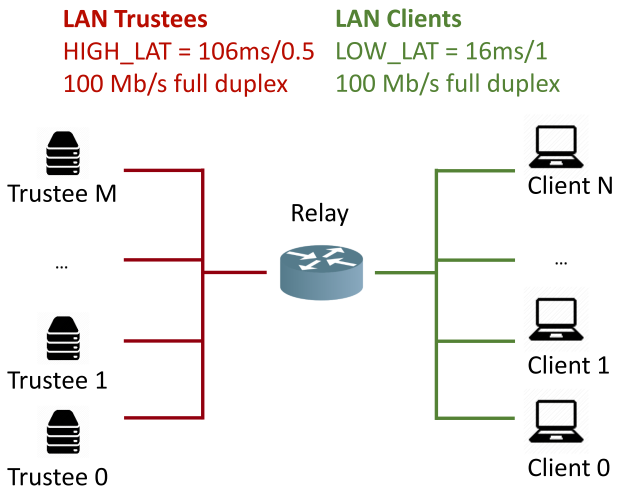


Figure 4.1: Network configuration

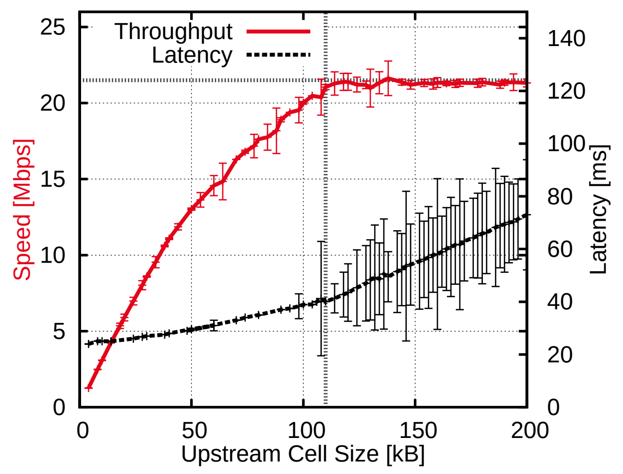


Figure 4.3: Effect of the cell size on the upstream bandwidth and the system latency.

5. Experiment results

Time to Resync. We want to evaluate the time when no upstream traffic can be processed, which happens when one client abruptly disconnects or times out. In other cases, like graceful disconnection, or new client connection, we theoretically could manage no *time-to-resync* by leveraging more computational resources: we keep the protocol Anonymize running, and we do run a new instance of Resync *in parallel*, stopping the old protocol only when the new one is ready to communicate. Still, we would like this protocol to be as fast as possible.

To measure the time-to-resync, we vary the number of clients and trustees, start measuring the time when the upstream traffic *stops being processed*, and stop measuring when the traffic restarts. We start with 2 clients, and progressively add clients; this also handles the case of client disconnection, as the resync protocol only depends on the final number of clients (e.g. adding 1 client to a setup with n clients result in the same time-to-resync than 1 disconnection in a setup of $n+2$ clients).

Results are visible in figure 2(a). We see that the time-to-resync increases both with the number of clients and trustees. For a given number of clients, we see that the number of trustees has a noticeable impact: this is because the oblivious shuffle used in round scheduling is *inherently sequential*, that is, each trustee does an operation in turn.

We see that for a given number of trustees, an increasing number of clients has only a moderate impact on the time-to-resync, meaning that this part of the protocol scales well with the number of clients.

Additionally, we optimize the protocol by making it more parallel. In practice, it means that connecting to any relay (and exchanging public parameters such as: the cell size, the number of clients, etc.), and collecting the public keys from the clients now happens in parallel. Results are displayed in figure 2(b). In the optimized case, the time is mostly spent on the network communication with the trustees, and is now close to the theoretic minimum.

Throughput and latency. We evaluate the total upstream, anonymized throughput of the protocol when varying the *upstream cell size*, or the number of bytes sent per round and per client. To test the system capacity, we run 1 client and 3 trustees, and measure the upstream speed this client gets. In addition, we measure the latency (defined as the round-trip time from the client, to the relay, and back to the client).

Figure 4.3 shows that we obtain more throughput with bigger upstream cell sizes; we almost reach 21 Mbps of throughput, along with the decent latency of 40ms.

After a cellsize of 110 KB, not only does the throughput stop increasing, but the latency increases drastically, as do the variance of the latency; this is the point of saturation of our protocol.

Window size. In PriFi, upstream and downstream traffic are *not* independent: clients need to make sure that a malicious relay is not trying to distinguish them by sending different pieces of information to different clients. For that purpose, clients usually share a hash of the history of the received messages. This means that the default way to send up and down cells is in lock-step: one cell upstream, one cell downstream, and so on. This is a terrible approach when trying to maximize the throughput, especially in a network where the bandwidth available is high, and the latency is moderate or high.

To palliate to this problem, we introduce a *window mechanism*, akin to what TCP does; in essence, for a window $W = 2$, we allow two cells to be going upstream for one going downstream. When the first upstream cell is acknowledged, the third one is sent, and so on.

As we see in Figure 1(a), the default lock-step mechanism (equivalent to $W = 1$) performs terribly, and increasing the window drastically improves the downstream speed. In our current setup, we almost saturate the 100Mbps link with a cell size of 30KB and a window of 10, whereas only 7Mbps were exploitable for a window of 1 and the same cell size.

Figure 1(b) shows that the latency (defined as the round-trip time from the client to the relay, through our protocol) only grows moderately with W, and remains acceptable for most applications for the tested values of W.

We emphasize the trade-off between bandwidth and latency visible when comparing Figures 1(a) and 1(b); furthermore, we point out that W could be *dynamically tuned* by the relay without interrupting the protocol, thus allowing users to adjust the bandwidth and latency following their needs.

6. Discussions

The experiment shows that in the network setup specified (Figure 3.1, 4.1), we achieve the reasonably high bandwidth of 20 Mbps per second, while maintaining an acceptable latency of 40 ms (Figure 4.3).

We explored the Resync protocol, which is a sensitive point in the PriFi system (recall that someone disconnecting abruptly *prevents the others from communicating*). In Figure 4.2, we see that the protocol scales well in the number of clients, and we argued that the result is mostly dependent on the network links to the trustees.

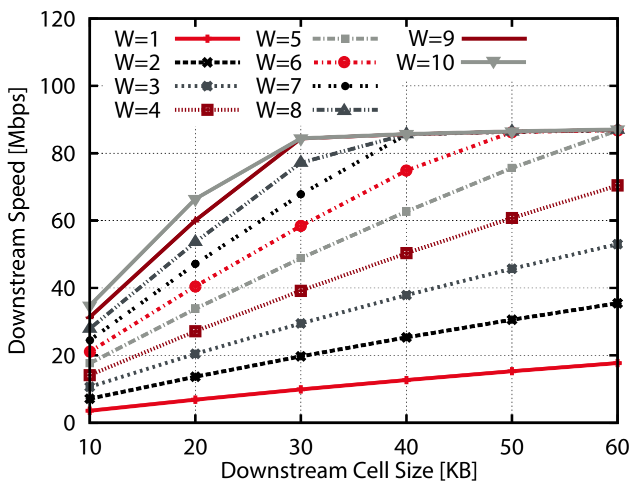
Furthermore, by introducing a window mechanism (Figure 5.1), we see that we are able to process downstream traffic at around 85 Mbps, close to the network limit of 100 Mbps; hence, given the appropriate parameters, we can make sure that PriFi is not the bottleneck on the global downstream (e.g. a video stream from a web server). Recall however that if the bandwidth from the trustees to the relay is limited, there is a trade-off between the upstream anonymized throughput (that requires M times the bandwidth from the trustees to the relay) and the downstream throughput.

7. Applications

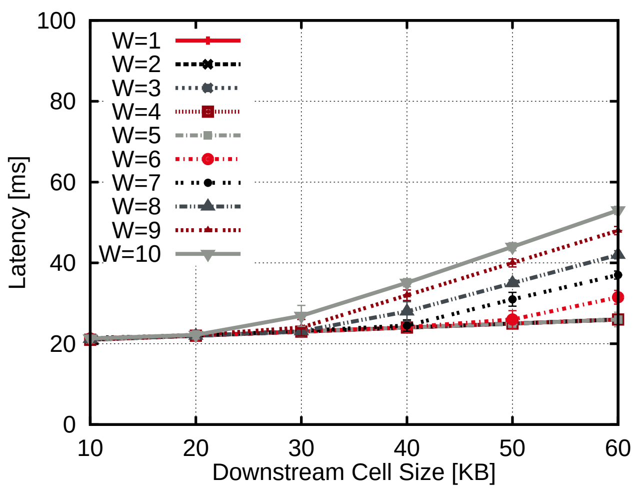
IoT and home protection The Internet of Things and the new connected objects in house appliances bring new threats to privacy. By eavesdropping on a wireless home network (even without being authenticated on the network), an attacker can gain substantial information about the habits of the inhabitants and their current whereabouts [2, 31]. Hence, it is important to hide the communication patterns of the devices (as long as the performances are not too hindered by this extra layer of security).

PriFi would be particularly well-suited in this setup with IoT devices, as it is not CPU-intensive (no heavy cryptography in the communication phase). In addition, IoT devices are usually not communication-intensive, allowing PriFi to be really lightweight[4], while still giving the strongest anonymization guarantees.

[4]In PriFi, the total communication is the product between the number of non-ciphertext bits transmitted, and the number of parties in the DC network. If the amount of data transmitted is modest, the CPU and bandwidth cost imposed by PriFi is also modest.



(a) Varying the window W has a tremendous impact on the downstream speed.



(b) Increasing the window W has a moderate impact on the system's latency.

Figure 5.1: Experiments related to the window size W.

Company & Bank network We explore a well-known attack, often referred to as the *parking lot attack*[5]. Such attacks are concrete threats to sensitive workplaces, such as banks: in practice, most of them do not have a wireless network for this reason, and forbid employees to create hotspots.

The threat is that an attacker, eavesdropping on the (encrypted) wireless network, may infer information about the communication coming from specific devices (akin the "IoT and home protection" scenario, but with threats not limited to user privacy).

We go further and also consider the case of a *rogue employee attack*. In large corporations, it becomes hard to fully trust every employee; the possibility of an employee using its credentials, hence being authenticated on the network, to eavesdrop upon other employees is a very concrete threat. Recall that, for instance, using TLS does not protect against this attack, as it does not hide the target website; we argue that the access patterns to various internet resources can be highly sensitive, leaking precious information about the employees' current tasks.

The current common security tools do not include ways to protect against this threat; only security-conscious people might use Tor as a personal initiative, but it is usually not deployed company-wide for every employee, also for accountability reasons. The only current way of mitigating this threat is to clusterize the network, and this solution has a significant cost.

Using PriFi in this setup would provide an important extra layer in the security stack, protecting against both parking lot and insider attack, giving strong guarantees that, for instance, the CEO cannot be spied upon by a rogue employee.

It is a long-term (given the perfect-anonymity protection), low-cost solution (clients only need to use a VPN-like application), and it can be vastly outsourced by the use of public or external trustees (We imagine that on the long term, external parties could *sell* trust, by running a trustee for a specific company).

In addition, it is possible to have accountability in PriFi: if every trustee agrees, the oblivious permutation used by the client could be opened, allowing to efficiently de-anonymize the clients, should the need arise. Typically, we imagine this scenario being used when a company is legally compelled to do so.

Mobility In the current setup, a roaming businessman can connect to the PriFi network of his company, and still get all the anonymity guarantees; the downside is that users of PriFi inside the company now devote more resources only to anonymize the roaming user (which might be an acceptable solution in most cases).

In the long run, we imagine supporting mobility of clients in a better way: We imagine a "roaming anonymous network", similar to Eduroam [11], that could be deployed all over the world; these instances of PriFi would be configured to use certain public trustees. Clients willing to have an anonymous connection could check the trustees list, and as long as they trust at least *one* of those, would be confident that the anonymity properties are respected. In addition, performance could be optimized by selecting close-by trustees.

[5]An attacker sets up a high-gain Wifi antenna in the parking lot of a company, and eavesdrop on the wireless network

8. Related Work

We divide the related work into two broad categories: anonymous communication protocols that forward data on a *best-effort basis*, thus usually yielding a good latency, but weak protection (if any) against traffic analysis; the second category being *fixed-time store-and-forward*, that intentionally delay traffic or wait on some condition before transmitting data, usually giving poorer latency performances, but stronger resistance against traffic analysis.

Best-effort forwarding. In the first category, the most well-known system is Tor [27]. Widely used, largely covered by media for the misuses and derives (Silk road [33] and other hidden services), with a good scalability, it is certainly the default choice for anonymous communication; as explained, it does not protect against global adversary, which we know exists [16]. The second most used tool to reduce the possibility of an eavesdropper is the VPN, that allow to bypass an eavesdropping point with the help of a relay, however also useless in the case of a global adversary that can see the entry and the exit point. HORNET [6] uses *onion routing* like Tor, but leverages on novel network tools to improve performance. Also sharing similarities with Tor, I2P [36] uses *garlic routing* instead of onion routing, and improves both performance using packet switching, and security using separate forward and backward routes. Intended to be much more lightweight than Tor, and protecting against a weaker adversary, LAP [18] boasts near-optimal delay and good performances by using stateless router protocols, and forwarding-state encryption. Finally, Crowds [29] relies on a probabilistic approach to forward the packets in a group of users before sending it to the destination.

As explained, PriFi does not fit in this category, as it aims to give stronger guarantees (which comes at a significant cost).

Fixed-time store-and-forward. In the second category, the best-known systems are without a doubt *mix-network* that intentionally delay traffic in order to resist against traffic-analysis attacks; the Chaumian mix [5], the Mix-Master protocol [32], and the Mixminion [8] system all fall in that category. Those systems in particular are not designed to have a low-latency; data can wait hours in the mixer until the size of the anonymity set is big enough. To palliate this problem, several systems include *chaff traffic*, or dummy traffic, in order to give a higher bound on the latency; it uses more bandwidth in order to give latency. In that category, Tarzan [12] is a fully-decentralized mix-net with chaff traffic, and uses IP domains to limit Sybil attacks; this is probably the work that is closest to PriFi, as they are both application-agnostic, traffic-analysis resistant protocols. However, Tarzan uses a peer-to-peer, more scalable protocol, whereas PriFi leverages on a specific network structure to give better latency performances.

Other recent work focused more on application-specific anonymous network, such as Aqua [23], tailored for anonymous P2P file sharing, and Herd [22], designed for anonymous VoIP. Generally speaking, DC-nets also fall in this category: Herbivore [14] is a file-sharing protocol that divides the set of users into groups, and constructs several DC-nets organized in rings, or cliques. Previous work on Dissent [7, 35] follows the same path of clustering users to improve performance, but aims to stay application-agnostic.

9. Conclusion and Future work

In this work, we introduced PriFi, an anonymous communication network tailored for wireless LANs and wired LANs. While exploring this new way of using DC-nets, we showed that performance was good enough to be used in real-life applications, with a reasonably high throughput and low-latency.

Future work. Generally speaking, there are several directions we would like to explore in order to *improve performance*:

1. Perhaps the most important improvement to be done is to use *UDP broadcast* from the relay to the clients instead of N-unicast TCP, thus leveraging the "free" broadcast given by a WLAN.
2. In a longer term, it would be interesting to study the feasibility of using the WLAN wireless channels to perform the XOR function now performed by the router. With the proper timing, we can construct a XOR function by having packets collide in the airwaves, and produce the result (instead of an unusable message like today's unwanted collision produce).

Those two changes should provide substantial performance improvements in terms of throughput and latency.

In addition, experimentation of PriFi on a wireless network needs to be performed; the current setup is purely wired, with Full Duplex links, and does not model the wireless behavior.

Finally, it would be interesting to include the verifiable component of the DC-net presented in [7] in PriFi, and measure the impact on the throughput and the latency; the interactions with the window mechanism should be particularly interesting.

Bibliography

[1] Matthew M. Aid. Nsa surveillance programs cover 75% of internet traffic transiting u.s., 2013. URL: http://www.matthewaid.com/post/58904880659/nsa-surveillance-programs-cover-75-of-internet.

[2] Sasikanth Avancha, Amit Baxi, and David Kotz. Privacy in mobile technology for personal healthcare. *ACM Computing Surveys (CSUR)*, 45(1):3, 2012.

[3] Laura Poitras Barton Gellman. U.s., british intelligence mining data from nine u.s. internet companies in broad secret program, 2013. URL: https://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html.

[4] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.

[5] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[6] Chen Chen, Daniele E Asoni, David Barrera, George Danezis, and Adrain Perrig. Hornet: high-speed onion routing at the network layer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1441–1454. ACM, 2015.

[7] Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 340–350. ACM, 2010.

[8] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 2–15. IEEE, 2003.

[9] Darlene Storm Darlene Storm. 75surveillance, 2014. URL: http://www.computerworld.com/article/2870773/75-of-writers-in-free-countries-self-censor-due-to-fears-of-mass-surveillance.html.

[10] DeterLab. Deterlab: Cyber-defense technology experimental research laboratory, 2016. URL: https://www.isi.deterlab.net.

[11] Global eduroam Governance Committee. eduroam, 2015. URL: https://www.eduroam.org/.

[12] Michael J Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206. ACM, 2002.

[13] Claire Davenport Georgina Prodhan. U.s. surveillance revelations deepen european fears of web giants, 2013. URL: http://www.reuters.com/article/europe-surveillance-prism-idUSL5N0EJ3G520130607.

[14] Sharad Goel, Mark Robson, Milo Polte, and Emin Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical report, Cornell University, 2003.

[15] Philippe Golle and Ari Juels. Dining cryptographers revisited. In *Advances in Cryptology-Eurocrypt 2004*, pages 456–473. Springer, 2004.

[16] Glenn Greenwald. Nsa collecting phone records of millions of verizon customers daily, 2013. URL: http://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order.

[17] Erik Hjelmvik. China's man-on-the-side attack on github, 2015. URL: http://www.netresec.com/?page=Blog&month=2015-03&post=China%27s-Man-on-the-Side-Attack-on-GitHub.

[18] Hsu-Chun Hsiao, TH-J Kim, Adrian Perrig, Akimasa Yamada, Samuel C Nelson, Marco Gruteser, and Wei Meng. Lap: Lightweight anonymity and privacy. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 506–520. IEEE, 2012.

[19] Google Inc. The go programming language, 2016. URL: https://golang.org/.

[20] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *Proceedings of the 2013*

ACM SIGSAC conference on Computer & communications security, pages 337–348. ACM, 2013.

[21] Benjamin Edelman Jonathan Zittrain. Empirical analysis of internet filtering in china, 2005. URL: `http://cyber.law.harvard.edu/filtering/china/appendix-tech.html`.

[22] Stevens Le Blond, David Choffnes, William Caldwell, Peter Druschel, and Nicholas Merritt. Herd: A scalable, traffic analysis resistant anonymity network for voip systems. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 639–652. ACM, 2015.

[23] Stevens Le Blond, David Choffnes, Wenxuan Zhou, Peter Druschel, Hitesh Ballani, and Paul Francis. Towards efficient traffic-analysis resistant anonymity networks. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 303–314. ACM, 2013.

[24] C Andrew Neff. Verifiable mixing (shuffling) of elgamal pairs. *VHTi Technical Document, http://www. votehere. net/vhti/documentation/egshuf. pdf, VoteHere, Inc*, 2003.

[25] NIST. Recommended elliptic curves for federal government use, 1999. URL: `http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf/`.

[26] The Tor Project. Traffic correlation using netflows, 2014. URL: `https://blog.torproject.org/blog/traffic-correlation-using-netflows`.

[27] The Tor Project. Tor project: Anonymity online, 2016. URL: `https://www.torproject.org/`.

[28] Xiao Qiang. How china's internet police control speech on the internet, 2008. URL: `http://www.rfa.org/english/commentaries/china_internet-11242008134108.html`.

[29] Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[30] Paul Syverson Roger Dingledine, Nick Mathewson. Tor: The second-generation onion router, 2004. URL: `https://svn.torproject.org/svn/projects/design-paper/tor-design.html`.

[31] Vijay Srinivasan, John Stankovic, and Kamin Whitehouse. Protecting your daily in-home activity information from a wireless snooping attack. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 202–211. ACM, 2008.

[32] P. Palfrader L. Sassaman U. Moeller, L. Cottrell. Mixmaster protocol version 2, 2004. URL: `https://tools.ietf.org/html/draft-sassaman-mixmaster-03`.

[33] Wired. The untold story of silk road, 2015. URL: `http://www.wired.com/2015/04/silk-road-1/`.

[34] David I Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Scalable anonymous group communication in the anytrust model. Technical report, DTIC Document, 2012.

[35] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *OSDI*, pages 179–182, 2012.

[36] Bassam Zantout and Ramzi Haraty. I2p data communication system. In *Proceedings of the tenth international conference on networks*, pages 401–409, 2011.