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

Ludovic Barman[†], Mahdi Zamani[‡*], Italo Dacosta[†], Joan Feigenbaum[‡],
Bryan Ford[†], Jean-Pierre Hubaux[†], David Wolinsky[§]

[†]EPFL, [‡]Yale University, [§]Facebook

ABSTRACT

Popular anonymity mechanisms such as Tor [4] provide low communication latency but are vulnerable to *traffic analysis attacks* that can de-anonymize users. Moreover, known traffic-analysis -resistant techniques such as Dissent [9] are impractical for use in latency-sensitive settings such as wireless networks. In this paper, we propose PriFi, a low-latency protocol for anonymous communication in local area networks that is provably secure against traffic analysis attacks. This allows members of an organization to access the Internet anonymously while they are on-site, via privacy-preserving WiFi networking, or off-site, via privacy-preserving virtual private networking (VPN).

PriFi reduces communication latency using a client/relay/server architecture in which a set of servers computes cryptographic material in parallel with the clients to minimize unnecessary communication latency. We also propose a technique for protecting against equivocation attacks, with which a malicious relay might de-anonymize clients. This is achieved without adding extra latency by encrypting client messages based on the history of all messages they have received so far. As a result, any equivocation attempt makes the communication unintelligible, preserving clients' anonymity while holding the servers accountable.

1. INTRODUCTION

Anonymous communication allows people to share information while being indistinguishable from other participants, and hence untraceable by an eavesdropping entity. Unfortunately, popular anonymous communication mechanisms, such as Tor [4], are not designed to protect against global surveillance, by which an eavesdropper can de-anonymize the users through traffic analysis attacks.

Conducting such attacks is especially easy in wireless local area networks (WLANs) common in organizations and homes. These networks are usually small and geographically local, allowing the eavesdropper to monitor the entire or a large portion of the network with a small resource cost. For example, in the so-called "parking lot attack," an adversary deploys low-cost equipment near a company's building to monitor encrypted WLAN communications, and to track or behaviorally analyze specific users and devices despite the WLAN's encryption. Such traffic analysis attacks are concrete threats to sensitive workplaces such as banks and defense organizations, which as a result sometimes choose a "no-wireless" policy, prohibiting wireless deployments of any kind inside their buildings.

Anonymous communication techniques could in principle help harden WLANs against traffic analysis attacks by making many users' and devices' network traffic indistinguishable. Current anonymity mechanisms, however, suffer from a tension between communication latency and traffic-analysis resistance: known techniques either focus on low latency and bandwidth costs while remaining vulnerable to traffic analysis [4, 5], or provide traffic-analysis resistance at the expense of high latency [6, 9].

This paper introduces PriFi, a low-latency anonymous communication protocol for local area networks with provable anonymity and resistance against traffic analysis attacks. PriFi is suitable for all-purpose communications similar to a VPN service but with strong privacy and tracking resistance. Users can communicate as usual via voice and video calls, text messages, or web browsing while ensuring their communication is anonymous and indistinguishable from the traffic of other users and devices on the WLAN. PriFi thus offers *local area anonymity*, where each user's identity is hidden among other users of the WLAN.

To achieve low communication latency, PriFi relies on a group of servers that are outside the latency-critical path taken by messages between the users and their Internet communication partners. To avoid adding latency, the off-path servers continuously compute cryptographic material that protects client anonymity but *does not* depend on the actual messages being sent, and thus can be computed and sent to the relay ahead of time. The clients' traffic is then aggregated, anonymized, and forwarded to the Internet by an untrusted relay node using the information sent earlier by the servers. This design allows PriFi to anonymize the traffic without adding extra hops to the latency-critical path.

Background. PriFi builds on Dissent [9], which is based on Dining Cryptographer's networks (*DC-nets*) [1] – anonymous communication protocols with provable traffic-analysis

*Corresponding author. Email: mahdi.zamani@yale.edu. Address: 51 Prospect Street, New Haven, CT 06515, USA

WPES'16, October 24 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2994620.2994623

resistance. DC-nets anonymize traffic by asking each participant to transmit equal-length *ciphertexts* synchronously.

To exchange a message, every pair of members shares a secret. Each member then produces its ciphertext by individually XORing its secret values together, with the anonymous sender additionally XORing in its message. Finally, all members broadcast their ciphertexts to other members. Since each secret is XORed into exactly two members' ciphertexts, all secrets cancel, revealing the anonymous sender's message without revealing the sender.

DC-nets require a scheduling mechanism to protect against collisions and a jamming-detection mechanism to prevent denial-of-service attacks. Such mechanisms often reduce the performance of the protocol significantly as, by default, every client has to share secrets with every other client. Each client must therefore compute and combine n secrets for every bit of shared-channel bandwidth, where n is the number of clients. Dissent alleviates this scalability issue by adapting DC-nets to a client/server model, in which the clients only share secrets with a small set of servers rather than with all other clients. As long as at least one of the servers behaves honestly, the anonymous message can be reconstructed correctly.

Dissent's more scalable DC-nets toplogy has a major impact on latency, however: it requires several server-to-server rounds of communication to handle client churn, maintain integrity of messages, and ensure accountability of participants. The main technical contribution of PriFi is a new DC-nets protocol redesigned so that these remote servers, while adding to the security of all PriFi clients, do not add to their communication latency. End-to-end latency of connections between users and local or remote sites they are accessing is dominated purely by "single-hop" communication via the PriFi relay. Even if the servers are geographically dispersed around the world, their existence adds security but not latency, because they only "stream" ciphertexts to the relay from outside the latency-critical path.

Threat Model. Consider a set of n *clients* (or *users*), who want to access the Internet anonymously. The clients are connected to this network through a *relay*, or router, that can process normal TCP/IP traffic in addition to running our protocol (see Figure 1). We also introduce a set of m servers whose role is to assist the relay in the anonymization process. These servers may be distributed around the world to maximize diversity and collective trustworthiness.

We consider an adversary who controls the relay, up to $n-1$ of the clients, and up to $m-1$ of the servers. We assume these servers satisfy the requirements of the *anytrust model* [9]: At least one server is honest, and they are all (including the relay) highly available, but clients need not know or choose which server to trust. We assume that all nodes communicate using non-private but authenticated channels; the adversary can observe all such messages when they are sent. While end-to-end content privacy is important in practice, it is readily achieved via orthogonal, well-understood content-encryption mechanisms such as TLS.

The goal of PriFi is to provide anonymity in such a manner that no one inside or outside of an organization can track the communication or attribute individual messages to their senders and receivers who are the members of the organization. From the point of view of an off-site member who is using the organization's PriFi network to communicate remotely with other members or the Internet, PriFi is similar to a low-latency VPN service: it receives data from and sends data to the applications running on the user's computer. The relay acts as the other end of the VPN, relaying data between the Internet and the clients. However, unlike traditional VPN services, the relay is not trusted; it may maliciously (possibly by colluding with other untrusted entities) attempt to de-anonymize the clients. The anytrust group of servers will collectively allow PriFi to protect the clients from de-anonymization by the VPN service, without adding latency into the critical communication path.

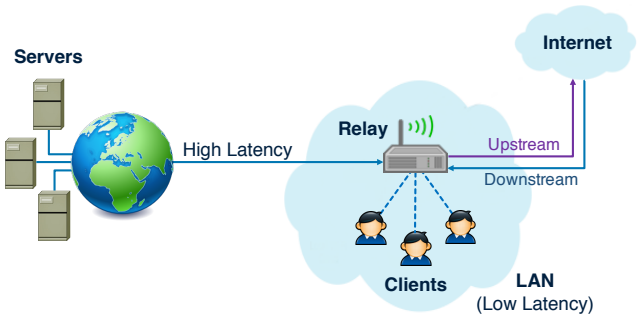

Figure 1: System setup

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.

2. SYSTEM OVERVIEW

Algorithm 1 summarizes the steps of a run of PriFi. These steps are executed jointly by the clients, servers, and the relay to anonymize messages sent by the clients to the Internet. The protocol proceeds in *time slots* such that only one client – the slot *owner* – is allowed to send an ℓ-bit anonymous message to the Internet in each time slot. A *round* is comprised of n time slots in each of which one of the clients becomes the slot owner and sends its next ℓ bits of data anonymously. PriFi only anonymizes the upstream traffic; the downstream messages are simply broadcast by the relay to every client, trivially providing receiver anonymity assuming the relay does not equivocate. Section 2.2 will discuss a mechanism to protect clients from an equivocating relay.

Before the protocol starts, we assume each client and each server holds a public/private key pair, and the relay holds a roster of all the public keys. This allows the relay to verify the membership of the clients and verify the authenticity of all communication. We now describe the three main phases of Algorithm 1: *setup*, *scheduling*, and *anonymization*.

Setup Phase. In the setup phase, the clients are authenticated using their public keys. Then, each client agrees with each server on a shared secret, which is known to both of them but is secret from others. This secret is later used to seed a pseudorandom generator (PRG) to obtain a stream of pseudorandom bits from which the clients and the servers will compute their ciphertexts. This allows the clients and the servers to avoid generating a shared secret for every slot.

Scheduling Phase. The scheduling phase determines which client gets to transmit its message in which slot. To achieve forward secrecy, each client generates an ephemeral public/private key pair that it uses for anonymization instead of its long-term keys. The ephemeral keys are refreshed in the scheduling phase at the start of each round.

Algorithm 1 PriFi

Notation. Let $\mathcal{C}_1, ..., \mathcal{C}_n$ denote the clients, $\mathcal{R}$ denote the relay, $\mathcal{S}_1, ..., \mathcal{S}_m$ denote the servers, and ℓ denote the bit-length of ciphertexts sent by clients and servers to the relay in each slot.

Setup

(1) $\mathcal{R}$ authenticates each client using its public key;
(2) For $i \in [n]$ and $j \in [m]$, each client $\mathcal{C}_i$ runs a key exchange protocol with each server $\mathcal{S}_j$ to agree on a shared secret $r_{ij} \in \{0,1\}^\ell$.

Scheduling

(1) Each client $\mathcal{C}_i$ generates an ephemeral pair of public/private keys (Z_i, z_i), sends Z_i to $\mathcal{R}$, and sets the time slot number $t \leftarrow 0$;
(2) $\mathcal{R}$ collects all Z_i's as a sequence A and sends it to $\mathcal{S}_1$;
(3) The servers take turn and shuffle A using [8] each sending its result and proof to other servers via $\mathcal{R}$;
(4) Each server verifies the shuffles, signs them with its private key, and sends the signature to all clients for verification.

Anonymization

(1) Each server $\mathcal{S}_j$ picks an ℓ-bit pseudorandom pad p_{ij} for each client $\mathcal{C}_i$ using a PRG seeded with r_{ij}. The server then computes its ciphertext
$$s_j \leftarrow p_{1j} \oplus ... \oplus p_{nj}$$
and sends it to $\mathcal{R}$;
(2) Each client $\mathcal{C}_i$ performs the following steps:
   a. Generate an ℓ-bit pseudorandom pad p_{ij} for each server $\mathcal{S}_j$ using a PRG seeded with r_{ij};
   b. Let π denote the permutation generated by the scheduling phase, and x_i denote $\mathcal{C}_i$'s next ℓ bits of data. Compute and send a ciphertext c_i to $\mathcal{R}$ such that if $t \bmod n = \pi(i)$, then
   $$c_i \leftarrow x_i \oplus p_{i1} \oplus ... \oplus p_{im}.$$
   Otherwise,
   $$c_i \leftarrow p_{i1} \oplus ... \oplus p_{im}.$$
   c. $t \leftarrow t + 1$;
(3) $\mathcal{R}$ collects the ciphertexts $s_1, ..., s_m$ from the servers and $c_1, ..., c_n$ from the clients and computes
$$y \leftarrow s_1 \oplus ... \oplus s_m \oplus c_1 \oplus ... \oplus c_n.$$
It then sends y to the corresponding Internet address;
(4) Upon receiving a response from the Internet, $\mathcal{R}$ broadcasts the message to all clients;
(5) If any client or server disconnects, $\mathcal{R}$ broadcasts a Reschedule request to all clients and servers;
(6) If any client or server receives a Reschedule request from $\mathcal{R}$, the receiver repeats the Scheduling phase.

The scheduling information needs to remain secret from all participants, as otherwise it can break the anonymity of all clients. In PriFi, the servers randomly and verifiably shuffle (using [8]) the sequence of ephemeral public keys corresponding to the clients. The secret permutation is then sent to all clients each of whom is only able to recognize its own public key in the sequence; other keys reveal no information to anyone without the associated private key.

Anonymization Phase. Each server continuously computes random ciphertexts for each client and sends them to the relay. Each ciphertext consists of ℓ random bits generated using a PRG seeded with the secret that the server shares with each client and is generated in the setup phase.

In every slot, each client also computes an ℓ-bit ciphertext and sends it to the relay; one of the clients sends a ciphertext that represents a meaningful ℓ-bit message that it wants to send to the Internet, and the rest send ciphertexts that represent *cover messages* consisting of ℓ zero bits. To produce an ℓ-bit ciphertext from an ℓ-bit message, the client first computes, for each server, an ℓ-bit *pseudorandom pad* using a PRG seeded with the secret that it shares with that server. Then, it computes its own ciphertext by XORing its ℓ-bit message with all of the pseudorandom pads.

The anonymization phase is repeated several times, and the clients take turns sending their messages in a round-robin fashion based on the shuffling information computed in the scheduling phase: if the current slot number modulo n points to the client's public key in the shuffled sequence of all public keys, then the client includes ℓ bits of its data in its upstream message for this slot. Otherwise, it just XORs the servers' ciphertexts and sends the result to the relay. Once the relay receives ciphertexts from all clients and servers, it XORs them together to obtain ℓ data bits. The data bits received from multiple rounds are combined together to form a valid TCP packet that is sent to the corresponding web destination. Upon receipt of a response from the Internet, the relay broadcasts the response to all clients.

2.1 Pre-Computing Server Ciphertexts

The server ciphertexts are generated by XORing pseudorandom values that are seeded with the secret each server shares with each client. Therefore, these ciphertexts do not depend on the actual communicated content. As a result, the servers can compute their ciphertexts ahead of the time before they are needed by the clients and the relay in the actual communication. This eliminates an important latency bottleneck from the critical latency path of the protocol. The servers continuously transmit freshly-produced ciphertexts to the relay throughout a round. Assuming that the servers have high-throughput links to the relay, the arrival of their ciphertexts outpaces the exchanges being performed by the clients and the relay, avoiding added latency.

2.2 Equivocation Protection

PriFi's client/relay/server topology introduces an important challenge: the untrusted relay can preform equivocation attacks by sending different (inconsistent) downstream messages to the clients to de-anonymize them. For example, in an unencrypted communication, the relay can slightly modify the downstream message for each client, thereby sending a unique message to each of them. These unique messages affect the requests that clients send in subsequent rounds; so the relay may be able, in these subsequent rounds, to determine which client sent each request.

PriFi protects clients from equivocation without adding extra latency by encrypting clients' upstream messages in such a way that the ciphertexts depend on all previous downstream messages. The relay will only be able to decrypt upstream messages if all clients agree on what they received in all previous downstream rounds; if they disagree, the XOR of all of the upstream messages that the relay receives will be garbage instead of the meaningful message encrypted by the slot owner. This allows the clients to ensure they have received the same message without imposing the latency overhead of a consensus protocol.

2.3 Further Improvements

1. Even if an adversary cannot directly track a client, it can run intersection attacks by guessing based upon online clients, especially if members of an organization must identify themselves to access the network. PriFi mitigates such attacks by authenticating clients anonymously using linkable ring signatures [7] with rotating linkage contexts. An attacker thus cannot tell which users are online at a given moment and cannot link anonymous users across rounds.
2. A malicious client can render the channel useless by continuously transmitting in every slot causing an untraceable denial-of-service attack. To address this weakness, PriFi redesigns Dissent's accountability mechanism to be compatible with PriFi's low-latency protocol.
3. Rather than incorporating all clients into the same PriFi communication channel, clients can subscribe to different channels with different bit rates. This allows PriFi to support clients with different energy/bandwidth constraints.
4. A single participant disconnection invalidates all ciphertexts in the anonymization phase. In most situations, however, the set of online clients remains stable enough to last for several exchanges. Thus, the servers update the set of online clients at regularly occurring reconfiguration events. The production of these configurations can be pipelined to reduce latency further.

3. PRELIMINARY RESULTS

We implemented a prototype of PriFi in Go and simulated its traffic and latency overhead using the CRAWDAD dataset [3]. The traces contain about 4,000 IP addresses with highly bursty and intermittent behavior; in particular, over the 4 hours captured by the traces, most devices only appear for a few minutes, before disconnecting forever. Out of these 4,000 IP's, we decided that the 10 highest-volume devices would run PriFi (the cost of using PriFi typically increases with the presence of high-volume traffic), and we manually assigned them to PriFi clients.

We are interested in measuring the increase in bandwidth and latency of clients' upstream traffic when it is anonymized by PriFi compared to when no anonymization mechanism is in use. Depending on energy requirements of client devices, the protocol may choose to adjust the number of rounds per second. Therefore, we vary the number of rounds per second between 0 and 200 and compute ℓ, the length of ciphertexts, such that the bandwidth used does not exceed the network capacity. To better match the downstream/upstream ratio usually found in web applications, we fix the downstream bandwidth to be at most 10 times higher than the upstream bandwidth.

Figure 2 shows the simulation results. The blue plot in the left figure depicts the increase in the number of upstream messages sent by every client, and the red plot depicts the increase in the number of upstream bytes sent by every client. In the right figure, the added latency is shown for the same experiment. These plots show the trade-off between latency and number of messages transmitted; the former decreases when the latter increases. We see that the number of bytes increases up to a limit, which is linked to the base traffic and the number of users. The spike around 90 rounds/sec is due to the size of the PriFi cell becoming smaller than the

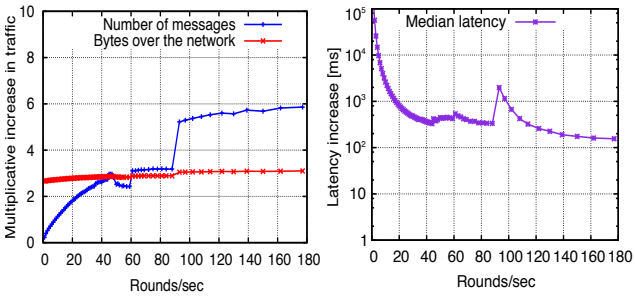

Figure 2: Upstream bandwidth and latency increase

maximum transmission unit in the traces: each message is being split in two, doubling the number of messages sent, and adding extra latency. Moreover, for 85 rounds/sec, we add latency in the order of magnitude of 100ms, for an increase of 3 times the number of messages, and 3 times the number of bytes sent.

4. CONCLUSION AND FUTURE WORK

We introduced PriFi, an anonymous communication network tailored to 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.

In the future, it would be interesting to study the feasibility of using network coding techniques to reduce latency further: e.g., can the wireless channel perform the XOR functionality currently performed by the router? Moreover, PriFi should be tested on a wireless network; the current setup is purely wired, with full duplex links, and does not model the wireless behavior. We also plan to use UDP broadcast from the relay to the clients instead of n-unicast TCP, thus leveraging the "free" broadcast given by a wireless LAN. Finally, it would be interesting to include the verifiable component of the DC-net presented in [2] in PriFi, and measure the impact on the throughput and the latency.

Acknowledgements. This research was supported in part by NSF grants CNS-1407454 and CNS-1409599, DHS grant FA8750-16-2-0034, William and Flora Hewlett Foundation grant 2016-3834, and by the AXA Research Fund.

5. REFERENCES

[1] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.
[2] H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford. Proactively accountable anonymous messaging in Verdict. In *USENIX Security*, pages 147–162, 2013.
[3] CRAWDAD. Dataset of wireless LAN traffic, 2009. URL: http://crawdad.org/pdx/vwave/20090704.
[4] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *USENIX Security*, 2004.
[5] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *CCS*, pages 193–206, 2002.
[6] S. Goel, M. Robson, M. Polte, and E. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical report, Cornell University, 2003.
[7] J. K. Liu and D. S. Wong. Linkable ring signatures: Security models and new schemes. In *ICCSA*, 2005.
[8] C. A. Neff. Verifiable mixing (shuffling) of ElGamal pairs. *VHTi Technical Document, VoteHere, Inc.*, 2003.
[9] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in numbers: Making strong anonymity scale. In *OSDI*, 2012.