Transparent Low-Latency Network Anonymisation for Mobile Devices

Martin Byrenheid(✉), Stefan Köpsell, Alexander Naumenko, and Thorsten Strufe

Chair of Privacy and Data Security, Technische Universität Dresden, Dresden, Germany
{martin.byrenheid,stefan.koepsell,thorsten.strufe}@tu-dresden.de, alexnau@posteo.de

Abstract. Mobile devices such as smartphones and tablets have become increasingly popular tools for Internet-based communication such as web browsing and text messaging. At the same time however, mobile devices fail to provide important privacy guarantees for their users. In particular, mobile devices per default neither conceal which services they are contacting nor hide their source IP addresses. Solutions to these problems exist, but either do not provide sufficient protection or have not gained widespread use due to a lack of usability. In this paper, we therefore present an architecture that combines the transparent tunneling of traffic with the strong protection of low-latency anonymisation networks. We furthermore present and discuss trade-offs that can be made to reduce the latency and overhead caused by the transparent tunneling of traffic. Based on measurements taken from a testbed setup, we show that our solution provides anonymity at the IP layer with acceptable energy consumption and goodput penalties.

1 Introduction

Wireless and mobile communication technologies such as Wi-Fi, UMTS and LTE allow mobile connectivity to the Internet at speeds sufficient for email, instant messaging, web browsing, and even video streaming. As a consequence, mobile devices such as smartphones and tablets became widely used for Internet-based communication.

Similar to desktop devices, smartphones and tablets per default do not provide sufficient concealment of the users communication activities. On the one hand, mobile devices do not hide the IP addresses of the services they are contacting, consequently leaking this information to the Internet service provider (ISP) and to local Wi-Fi access point providers. On the other hand, mobile devices do not hide their own IP addresses from the services they are contacting. Thus if the service colludes with the ISP of the user, both entities are able to link the service with the real-world identity of the user.

R. Beyah et al. (Eds.): SecureComm 2018, LNICST 254, pp. 193–209, 2018.
https://doi.org/10.1007/978-3-030-01701-9_11

Virtual Private Network (VPN) services like Mullvad[1] and Private Internet Access[2] allow users to mitigate these problems by installing an application on their devices that tunnels all traffic through the VPN provider's servers. While VPN services typically act transparently and can be set up easily, they require the user to trust the VPN provider, since the latter can easily observe which services the user is contacting [1,2]. Applications like Orbot[3] and ANONDroid[4] in contrast utilise low-latency anonymisation networks and thus provide anonymisation without the user having to trust a single entity. At the same time however, these applications do not yet provide the same level of transparency as VPN services do. The ANONdroid application only provides a local SOCKS-interface, so that only applications that support SOCKS can profit from anonymization. Orbot already provides a VPN mode that achieves a similar level of transparency as applications from VPN providers do. However, Orbot only supports anonymisation of TCP traffic[5] and we are not aware of any documentation or evaluation of Orbot's architecture and performance.

Motivated by the need for easy-to-use and strong IP layer anonymisation, we present the following contributions in this paper:

- Inspired by virtual private network technology, we propose an architecture for transparent tunneling of arbitrary IP traffic through low-latency anonymisation. We furthermore discuss protocol-specific optimisations for UDP, TCP and DNS traffic.
- We present results from experiments on a testbed, where we measured the average goodput and energy consumption during the download of data under different settings. In particular, we compare the performance of different devices with different versions of Android for the cases that OpenVPN for Android, Orbot or our solution is used for anonymisation.

The paper is structured as follows: In the next section, we first give a short introduction to mix networks and highlight how low-latency mix networks differ from traditional mix networks. In Sects. 3 and 4, we state the goals that guided the design of our solution and discuss to which extend these goals have been addressed in the literature. In Sects. 5 and 6, we subsequently present our solution in detail and discuss protocol-specific optimisations. Section 7 discusses to which extend the optimisations we proposed weaken the privacy guarantees of the low-latency anonymisation service that is used. In Sect. 8, we present and discuss our experimental results. Section 9 then provides summary of our results.

[1] Mullvad. https://www.mullvad.net/en/, 2018-01-05.

[2] Private Internet Access. https://www.privateinternetaccess.com/. 2018-01-05.

[3] "Orbot: Tor for Android". Guardian Project. https://guardianproject.info/apps/orbot/. 2018-01-05.

[4] "ANONdroid". JAP-Team. https://play.google.com/. 2018-01-23.

[5] "Tor Project: FAQ". Tor Project. https://www.torproject.org/docs/faq.html.en. 2018-01-05.

2 Background

Our work addresses privacy issues of mobile devices such as smartphones and tablets that are connected to the Internet via Wi-Fi or mobile communications. We consider a user, who runs one or more applications on his device. Each application in turn contacts a number of Internet services to exchange data. Depending on the service, the respective application might need to provide login credentials in order to gain access.

To protect his privacy, the user wants to set up his mobile device in such a way that the installed applications never contact Internet services directly, but instead utilise an anonymisation service like Tor [3] or AN.ON [4] to contact services privately. In particular, the user expects the service to provide *sender anonymity*, meaning packets sent by the user's device cannot be linked to it anymore after being processed by the network.

Low-latency mix networks are based on the mix concept introduced by Chaum [5]. To conceal its communication using a sequence of mix nodes, a client pads each packet to a fixed length and performs layered encryption, where each layer is encrypted with the key of the corresponding mix node. Low-latency mix networks are designed for use cases where a high latency results in unacceptable slowdown, such as web browsing or multimedia streaming. Instead of encrypting each mix packet separately using public key cryptography, low-latency mix networks therefore only make use of public key cryptography once to exchange symmetric keys between senders and mixes. A set of symmetric keys is then called an *anonymous channel* or *circuit* and is used for multiple packets. The low-latency mix networks considered in our work furthermore provide reliable communication, automatically performing retransmission of lost packets.

Due to the absence of any artificial delay during the processing of mix packets by mixes, low-latency mix networks are vulnerable to traffic analysis attacks based on timing. Thus, low-latency mix networks can only protect against a *local adversary* that can observe or compromise a fraction of the mixes by generating, modifying or dropping packets. Furthermore, low-latency mix networks do not provide anonymity for a particular channel if the first and last mix of the channel are under control of the adversary.

3 Goals

In our opinion, a useful and effective mechanism for mobile devices that provides anonymisation of traffic at the network and transport layer should achieve the following goals:

1. **Strong anonymisation of traffic:** The mechanism must support obfuscation of arbitrary traffic at the network and transport layer in a manner that does not open up new attacks for the local adversary described in Sect. 2.
2. **Low setup complexity:** The mechanism needs to be easy to setup on current operating systems for mobile devices. In particular, it shall be implementable without requiring modifications of the operating system kernel or

unlocking of administrative privileges (e.g. rooting on Android or jailbreaking on iOS). Furthermore, the solution should provide protection for all installed applications per default without the need to make changes to any application.
3. **Low overhead:** The anonymisation should not result in a significant loss of network throughput nor increase the latency significantly. Furthermore, the solution should consume a low amount of energy.

We consider anonymisation at the application layer out of scope, since latter cannot be done in general without changing the code of the applications themselves. For example, web browsers typically protect the confidentiality and integrity of their traffic by means of Transport Layer Security (TLS). Consequently, a transparent proxy application cannot erase identifying information from the web traffic without being able to compromise TLS encryption, which is clearly undesirable.

4 Related Work

According to our knowledge, the integration of low-latency anonymisation services into mobile devices so far received only few attention by the research community. Wiangsripanawan et al. [6] proposed game-based definitions for location privacy as well as for sender anonymity, receiver anonymity and unlinkability together with three different mechanisms that allow mobile Tor clients to avoid loss of existing circuits after their IP address have changed. Andersson and Panchenko [7] investigate the differences regarding anonymity and performance for the cases that the Tor client is running directly on the mobile device, the Tor client is running on a computer at the home of the user and the case that the mobile device connects to a third party Tor client. More recently, the work of Doswell [8] presents a field study along with network simulation and mathematical modelling, showing the impact of congestion and circuit build time on the achievable data transfer speed. Furthermore, Doswell proposes a design called *mBridge*, which employs a trusted bridge relay as first hop and Mobile IP [9] to handle changes of the mobile devices' IP address.

Fundamentally, all previous work on low-latency anonymisation that we are aware of focuses on efficient maintenance of connectivity to the anonymisation service. Thus, there still is a need for investigation regarding how an anonymisation client can be integrated into the mobile device in an effective and resource-efficient manner.

5 Design

Existing low-latency anonymisation networks like Tor [3] or AN.ON [4] typically provide a client software that is run as a background service on the user's system. In order to make the anonymous communication service accessible to other applications, the client software implements a proxy interface and can therefore be used like a regular proxy (usually SOCKS or HTTP proxy). While this approach has the advantage that it only requires processing of application layer data and is thus easy to implement, it currently fails in practice, as Android and iOS do not support a corresponding permanent global setting. Instead, Android and iOS only allow manual proxy settings per network [10,11], which would have to be done manually by the user and thus is error-prone.

To overcome the aforementioned problem, we instead utilise a virtual network interface (VNI), which is already provided by Android and iOS [12,13]. Latter is handled by the operating system like a real network interface. The interfaces provided by Android and iOS furthermore automatically set up their forwarding table so that whenever an application sends a packet to an arbitrary destination, this packet will be enqueued in the VNI until it is processed by a dedicated process that we call *mediator service* in the following.

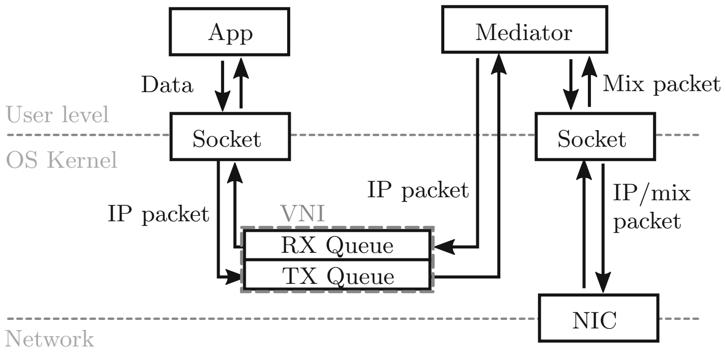

Fig. 1. The architecture of the software running on the user's smartphone.

Figure 1 illustrates the architecture of the software running on the user's device in detail. Except for the mediator service, all packets sent by user-level applications are appended to the VNIs *transmission queue* (TX queue) by the operating system. We assign an IP address from a private address space (10.0.0.0/8 or 172.16.0.0/12) to the VNI in order to avoid overlap with IP addresses of the Internet.

The only process that is able to exchange IP-packets with the external network is the mediator service. Latter is aware of the VNI and is able to read and remove IP-packets from the transmission queue. Additionally, the mediator service is able to append IP-packets to the *reception queue* (RX queue) of the VNI. The operating system subsequently delivers all IP-packets added to the reception queue to the corresponding applications.

For each IP-packet read from the transmission queue, the mediator service extracts the relevant header data and payload, sets up a new channel through the anonymisation service if necessary and submits the extracted data over the channel. Whenever the mediator service receives data from the anonymisation service, it assembles a corresponding IP-packet and appends it to the reception queue. If necessary, the mediator service then closes the channel through the anonymisation service.

5.1 Improving Efficiency and Anonymity Considering Transport Layer Packet Streams

Since we focus on current mobile devices based on Android and iOS, the primary consideration that influenced our design is that latter devices mainly communicate in the form of TCP and UDP flows, where each flow consists of multiple packets destined to the same recipient.

From the perspective of the mediator service introduced above, we assign each field of a packet header to one of the following three categories:

- **Static:** Data that needs to be transmitted and which does not change from packet to packet of the same flow.
- **Dynamic:** Data that needs to be transmitted and which differs between packets of the same flow.
- **Private:** Data that must not or does not need to be transmitted.

Additionally we treat the whole packet payload (i.e. the application layer data) as dynamic because our anonymisation approach works on the transport layer and has therefore no knowledge about which parts of the payload are in fact static or private.

For static data, we can trade off transmission overhead against memory consumption (in terms of keeping state). To do so, the mediator *only once* sends control information that contains the set of header fields and values that shall be used for all packets of the corresponding flow. The last relay then keeps this information in memory until the flow has been terminated.

Dynamic data in turn allows a trade-off between overhead and latency. Since the encapsulation process of the low-latency anonymisation protocol enforces a fixed length for the resulting mix packets, the dynamic data contained in one packet might only use a fraction of the space reserved for anonymous payload data, thus resulting in significant padding overhead. To reduce the needed amount of padding, the mediator can wait a certain amount of time for further packets and if successful, encapsulate the dynamic data of multiple packets within one mix packet.

In the following we describe how we adapted the general approach described above to the important transport layer protocols UDP and TCP and how we deal with the remaining IP traffic.

UDP: The UDP header contains four fields, namely the destination port, the source port, the payload length and a checksum. The destination port is considered to be static data and thus will be transmitted (together with the destination IP address) only once per UDP packet flow. Additionally we treat the source port as static data. The checksum is handled as private data and therefore not transmitted. As the anonymous channel offers reliable transport, the checksum will be recreated by the last relay before sending the UDP packet to the final destination. The UDP payload length is considered to be dynamic. We assume that many application layer protocols on top of UDP expect to receive non-fragmented data units (e.g. RTP packets) and thus we must preserve the UDP packet size while sending the UDP packet from the last relay to the final destination.

TCP: For TCP, the situation is more complex, since a naive transmission of TCP header data and payload would neglect the fact that low-latency anonymisation services already provide reliable transmission of data, thus resulting in an undesirable transport of TCP over TCP. Additionally it turned out that most of the TCP header fields could or should be treated as private. The former for efficiency reasons, the latter due to anonymity reasons, e.g. to avoid leakage of details about the sender's TCP implementation (which in turn could be related to revealing the operating system used etc.). Therefore our general approach is to basically only transport the TCP payload but not the TCP header. The only exception is that we treat the destination port number (and destination address) as static data.

To achieve this, we employ a minimal user-level TCP stack that allows the mediator service to transparently interact with the TCP streams of local applications. Since the mediator service and the application that uses TCP are running on the same device, the user-level stack does not need any sophisticated mechanisms for detection of packet loss and congestion control, allowing us to avoid high computational overhead. Figure 2 illustrates the interaction between the local application, the mediator service, the anonymisation service and the recipient upon initiation of a new TCP connection by the local application. Whenever the mediator service takes a TCP SYN packet from the transmission queue, it opens a new anonymous channel and sends a signalling packet with the corresponding IP address and port number to the last relay of the anonymisation service. Latter then initiates a TCP connection using the given address and port number and responds with a signalling message that indicates whether the connection has been set up successfully. If the connection succeeded, the mediator service generates a TCP packet for the local application with the SYN and ACK flag set. Otherwise, the mediator service generates a TCP packet with RST bit set and appends it to the reception queue.

During transmission of data by the local application, the mediator will transparently extract the payload from the packets and send it over the established anonymous channel. When the application terminates the connection by sending a packet with the FIN flag set, the mediator will signal the anonymisation service to close the associated connection. If the recipient terminates the connection, the anonymisation service will notify the mediator, which in turn will shut down the TCP connection of the corresponding application.

IP. Although the official APIs of Android and iOS currently only support UDP and TCP, we also discuss anonymisation solely at the IP layer in the following.

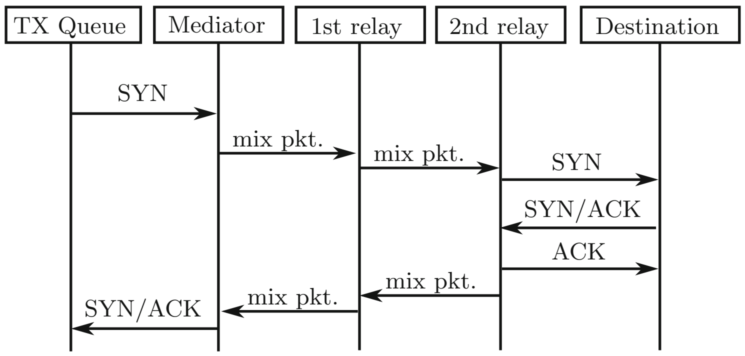

Fig. 2. Interaction between the local application, the mediator service and the anonymisation service for TCP transmissions (pessimistic mode of operation).

This allows us to provide at least a minimal protection in the case that other transport layer protocols are supported by Android or iOS in the future.

We treat all IP-traffic which is neither TCP nor UDP in the same way. First of all each IP-packet flow is transmitted over an individual channel to reduce linkability. For each IP-packet we send certain header information (as explained below) and the IP-packet payload. The last relay will create IP-packets out of this information and sends them to the final recipient.

Considering IP version 4 and IP version 6, the *Version* field, and the *Destination Address* clearly belong to the static data. The *Total Length/Payload Length* field belong to the dynamic data. The *Source Address* and the *Header Checksum* belong to private data.

Additionally, we categorise the *Differentiated Services Code Point (DSCP)* and *Explicit Congestion Notification (ECN)* as well as *Traffic Class* as private data. The reason is, that we are not aware of any low-latency anonymisation service which supports different service/traffic classes, thus limiting their effectiveness to only the path between the last relay and the recipient.

For IPv4 packets, we think that data related to fragmentation (*Identification*, *Flags*, *Fragment Offset*) can be treated as private data, if anonymity has priority. Fragmentation is rarely used in practice – and IPv6 does not even support it. In this case all fragmented IPv4 packets will be dropped. The *Time to Live (TTL)/Hop Limit* field can also be treated as private in case of prioritising anonymity. Therefore the last relay will set this field to a predefined fixed value. Nevertheless there might be cases where this field is used in protocols (traceroute would be one example). Therefore we consider it as dynamic data if compatibility needs to be prioritised.

The IP header supports optional headers by the *Options* and *IHL* fields in case of IPv4 and by the *Next Header* field in case of IPv6. Again, if anonymity should be prioritised, we consider these fields as private. More specific, we drop all IP-packets which have optional headers. If compatibility has priority, we do not change the fields and therefore consider them as dynamic data.

6 Optimistic DNS

In this section we will describe some efficiency improvements related to DNS, which we treat in a special way due to its importance.

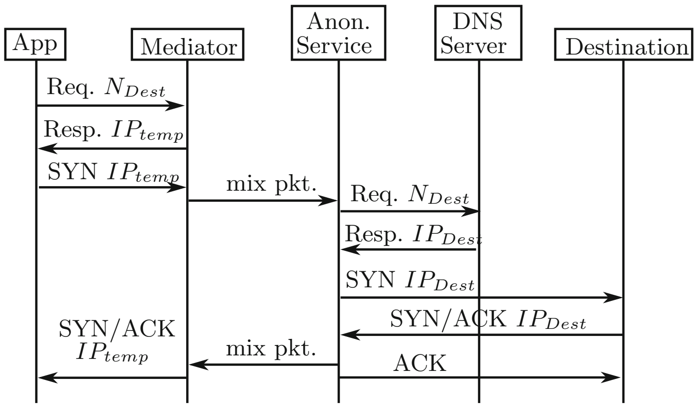

Fig. 3. Interaction between the local application, the mediator service and the anonymisation service for DNS requests.

One fundamental advantage of VNI-based anonymisation over SOCKS-based solutions is that the operating system can be configured to use the VNI for DNS name resolution. All DNS requests, including those triggered by applications, will then be appended to the transmission queue of the VNI, thus allowing the mediator service to perform name resolution over the anonymisation service. However, since DNS resolution needs to be completed before any packets can be sent to the intended recipient, we face the problem that the anonymisation of the DNS request by means of rerouting over multiple mixes introduces significant latency. We therefore propose an optimistic DNS resolution, which is illustrated in Fig. 3. Upon arrival of a DNS request, the mediator service immediately chooses a local, temporary IP address and generates a corresponding DNS response. The mapping from the temporary IP address to the DNS name is kept in memory by the mediator service, so that subsequent packets can be correctly associated with their destination. As soon as the mediator starts submission of data to the anonymisation service, it prepends signalling data containing the DNS hostname, leaving the task of DNS name resolution to the last relay.

In order to avoid that the DNS lookup table grows infinitely, we set the time-to-live (TTL) field in our DNS responses to 60 s and remove outdated DNS entries after 70 s. These timeout values reflect the default lifetime of application level DNS caches as used by Mozilla Firefox and Google Chrome.

7 Discussion

Considering TCP, we only transmit the minimal necessary data, namely the destination IP and port number and the payload itself. Since we do not see how to further reduce this information without breaking application layer compatibility we consider our TCP solution as optimal from a privacy point of view.

Considering UDP, our current implementation is close to the optimal case. The only field which we could possibly avoid to transmit is the source port number. In fact some preliminary tests suggest that letting the last relay choose a random source port will not break applications. But we need to do a more profound analysis here.

Considering our optimistic approach for DNS resolution, we see advantages and disadvantages with respect to privacy compared to the normal DNS resolution using anonymous UDP channels. Remember that in our optimistic DNS approach, the last relay eventually does the DNS resolution. Thereby it can link the payload of the related anonymous channel with the destination host name contained in the signalling packet. This in fact could leak more information compared to the case, where the last relay only learns the destination IP address (but not the host name). Think e.g. of a web server hosting multiple web sites. A common approach to distinguish the different web sites is based on the requested host name. Therefore the last relay will learn which of the multiple web sites was in fact requested (in case of optimistic DNS) while in case of normal DNS it will not learn this (assuming that in both cases the HTTP protocol itself is encrypted by TLS).

On the other hand, the optimistic DNS approach could also offer some privacy benefits compared to the normal anonymous DNS lookup. Assuming that there is a web-site https://gugle.com operated by Numbers Inc., the DNS servers of Numbers Inc. could respond with a new unique IP address for each DNS request for gugle.com. Usually DNS responses are cached on the client side, meaning that subsequent HTTP requests sent from the same client to gugle.com will all use the same destination IP address while HTTP requests originated from different clients will use different destination IP addresses. Therefore the web servers of Numbers Inc. can distinguish different users even if all of them use an anonymisation service for their communication (including DNS requests). In case of our optimistic DNS resolution, all DNS requests will be made by the same client, namely the last relay of the anonymisation service. Thus it will prevent the attack described above.

8 Evaluation

Network layer anonymisation on mobile devices is particularly challenging due to their constrained resources. A high computational overhead will result in low throughput and high battery drain, thus severely limiting the usefulness of the device. To evaluate the practical value of the design presented in Sect. 5, we implemented our solution (named *ANONguard*[6]) for the Android operating system and performed experiments on a local testbed[7]. We chose to evaluate our solution in a local testbed to obtain upper bounds on the performance of our solution in an idealized setting in which there is almost no network latency and congestion.

As depicted by Fig. 4, we set up a 802.11ac-capable wireless access point (AP) that allows a mobile device to connect with a dedicated measurement host. For our measurement, we used a Motorola Moto X Style running Android 7.0 and an ASUS Nexus 7 (2nd generation) running Android 6.0.1. Both devices were running the latest official version of Android that was provided by the manufacturer. To generate arbitrary traffic workloads, we implemented a measurement client for Android along with a measurement server application for Linux. The instrumentation host was used to perform repeated measurements in an automated fashion, as it controls the setup of the anonymisation software on the mobile device, starts the measurement client and extracts the measurement results from the device.

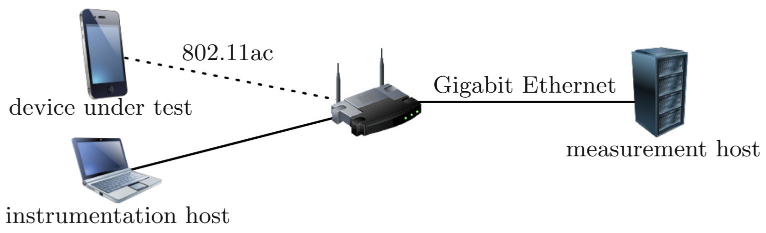

Fig. 4. The setup of the testbed used for experimental evaluation.

8.1 Goodput

In our first experiment, we investigated the loss of goodput due to network layer anonymisation. Clearly, we expect a certain loss of goodput due to the encapsulation of traffic into fixed-size packets that are smaller than the Maximum Transmission Unit (MTU) of the network card, which in itself does not necessitate measurements. However, even though many current smartphones have sufficient computational power, frequent multi-layer encryption and decryption might lead to high temperatures that force the device to slow down and thus limit the actually achievable bandwidth. Furthermore, the utilisation of a virtual network interface incurs additional computational overhead whose impact is hard to estimate analytically.

[6] A test version of ANONguard can be installed from Google Play (https://play.google.com/store/apps/details?id=anonvpn.anon_next.android).

[7] The source code for our evaluation tools as well as the used anonymisation tools can be found on https://dud-scm.inf.tu-dresden.de/ANON-Public.

Methodology: To obtain measurements, we first deployed a set of mixes on the measurement host and start the anonymisation software on the mobile device. Afterwards, we set up our measurement client to constantly download data from the measurement server as fast as possible and record the total number of bytes received within 10 min. To avoid errors due to instrumentation, the recording was started one minute after the download had began.

We performed measurements with one, two and three mixes to evaluate the impact of encryption and decryption on the achievable goodput, since each mix requires one decryption operation per incoming packet. In the case of one mix anonymisation, we used version 0.7.3 of OpenVPN for Android. For the two and three mix scenario, we used our solution as well as Orbot version 15.5.1-RC-2 (Tor version 0.3.1, compiled with Android NDK Rev. 15c). As reference, we also performed measurements without any anonymisation. Furthermore, we performed measurements where the client directly connects to our mixes and thus avoids any overhead associated with the usage of a virtual network interface.

Results: Figure 5 shows our results for different configurations and devices. Each dot represents the mean value of the average goodput, measured in megabit per second, over 10 runs and the bars denote the confidence intervals for a significance level of 95% based on Student's t-distribution.

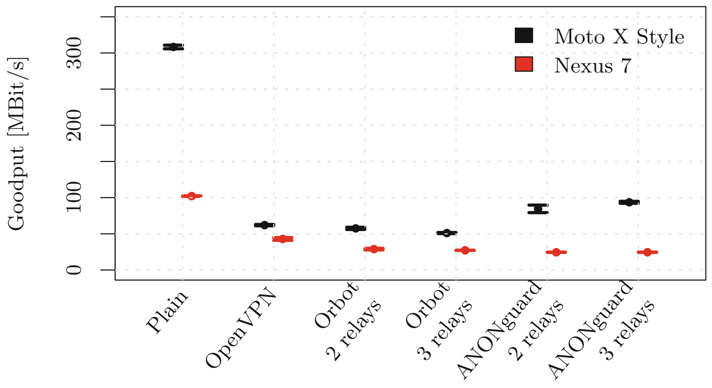

Fig. 5. Mean values for the average goodput during a continuous download for different devices and anonymisation solutions.

Clearly, the employment of current implementations of transparent tunneling leads to a significant drop in network performance. In the case of OpenVPN, the average goodput dropped from 308.2 ± 2.5 Mbit/s to 61.9 ± 0.9 Mbit/s on the Moto X Style and from 102.0 ± 0.5 Mbit/s to 42.8 ± 1.9 Mbit/s on the Nexus 7. Orbot achieved similar results, allowing 57.5 ± 1.5 Mbit/s on the Moto X and 28.8 ± 1.1 on the Nexus 7 if two relays are used and 51.0 ± 0.7 Mbit/s as well as 27.1 ± 0.1 Mbit/s if three relays are used. On the Moto X Style, the ANONguard application achieved a notably higher goodput than OpenVPN and Orbot, allowing 84.5 ± 5.2 Mbit/s if two relays are used. This is surprising, since the whole tunneling and anonymisation functionality of OpenVPN and Orbot is running as native code while ANONguard is purely written in Java. We experimented with different settings for OpenVPN and Orbot but were not able to substantially improve their performance. Another surprising observation is that the average goodput of ANONguard slightly improved to 93.5 ± 1.4 Mbit/s when three relays were used. Unfortunately, we did not find a definitive answer for this result in the time available for the measurement study.

While the Nexus 7 showed a rather continuous performance during each measurement run, performance on the Moto X dropped during each run where anonymisation was enabled. In all our results, we only included runs where the initial temperature of the Moto X was between 35 and 40° Celsius. During each run, the temperature of the device increased to a value close to 60° Celsius and then stabilised at a value around 53° Celsius after the CPU frequency has been automatically reduced. However, the drop in performance was not dramatic compared to the drop from goodput without anonymisation: in the first two minutes of the measurement, the average goodput of ANONguard with two relays reached 103.4 ± 2.5 Mbit/s whereas in the remaining 8 min, the average goodput reduced to 79.7 ± 6.8 Mbit/s. Similarly, Orbot's goodput dropped from 68.1 ± 1.4 Mbit/s to 54.9 ± 1.7 Mbit/s.

While the results shown on Fig. 5 have been obtained by a single TCP-connection, we performed the same measurements with 3 simultaneous TCP transmissions. In this setting, the average goodput without any anonymisation increased to 335.2 ± 16.3 Mbit/s, whereas the goodput using ANONguard increased to 98.9 ± 2.4 Mbit/s. We observed similarly low increases in goodput for OpenVPN and no difference for Orbot. Consequently, the number of simultaneous transmissions does not have a strong impact on our results.

Analysis of Goodput Degradation: To evaluate possible causes for the strong degradation of goodput, we performed additional measurements to asses the particular impact of encryption and decryption as well as interaction with the virtual network interface. The results presented by Fig. 5 indicate that encryption and decryption do not have a major impact on goodput, since the number of relays did not have a notable impact on the results of Orbot and ANONguard. To verify this claim, we ran the goodput measurement with a modified version of ANONguard where encryption and decryption has been disabled. Afterwards, we furthermore integrated the AN.ON client implementation that is used in the ANONguard application directly into the measurement client application, thus avoiding the overhead associated with the virtual network interface and conducted measurements without the ANONguard application. Even if cryptography as well as transparent tunneling is disabled, the AN.ON client just achieves 115.59 ± 10.50 Mbit/s of average goodput on the Moto X and 47.3 ± 0.6 Mbit/s on the Nexus 7. An initial runtime profiling indicates that internal synchronisation is a primary cause for the loss of performance of ANONguard. After we re-enabled encryption, the goodput slightly dropped to 108.1 ± 3.2 Mbit/s on the Moto X and 43.5 ± 4.5 Mbit/s on the Nexus 7. Using just transparent tunneling without encryption yielded a goodput of 83.6 ± 1.2 Mbit/s on the Moto X and 25.3 ± 0.9 Mbit/s on the Nexus 7. Besides internal synchronisation, profiling in latter case suggests that the frequent computation of TCP checksums contributes to the loss of performance.

Discussion: Our measurements suggest that current solutions for network layer anonymisation significantly reduce the achievable goodput on mobile devices. However, the observed loss of performance mostly seems to stem from networking-related performance issues such as synchronisation of concurrent threads as well as computation of checksums. We did not observe a strong additional loss of goodput after enabling the virtual network interface together with encryption and decryption, which indicates that our current results underestimate the actual goodput that is achievable if there are no networking-related performance bottlenecks.

8.2 Energy Consumption

Complementary to our measurements regarding goodput, our second experiment focuses on the energy consumption of the transparent anonymisation.

Methodology: To obtain samples, we performed the same steps as described in the previous section but before launching the measurement client, we disabled the power supply of the mobile device and used the Trepn Power Profiler[8] to record the battery power of the device every 100 ms (which is the highest frequency Trepn supports). If available, the Trepn Power Profiler reads battery power from the fuel gauge of the battery pack [14]. After 12 min have passed, we stop the recording of the battery power and use the mean value of all data points between the second and 12th minute as sample value. Before the next measurement run was started, we charged the battery of the device to 95%. Since the Motorola Moto X Style does not have an integrated fuel gauge, we only performed measurements with the Nexus 7, which comes with a BQ2751 fuel gauge from Texas Instruments (according to the kernel log messages of the device).

Our measurement procedure provides a worst case perspective on the increase in battery drain, since it seems unlikely that smartphones constantly have to cope with one or more running TCP transfers. However, we are not aware of any published models or datasets that allow us to emulate the variety of dynamic workloads that mobile devices typically have to deal with.

As a reference, we performed measurements without anonymisation on minimal screen brightness and also on maximum screen brightness. All measurements involving anonymisation were recorded with minimal screen brightness. The wakelock acquired by the Trepn profiler ensured that the device did not change its screen brightness or went to sleep mode during the transfer. We also recorded the average goodput of each run and observed a decrease of around 5 Mbit/s for the setting without anonymisation and the setting with OpenVPN.

[8] https://developer.qualcomm.com/software/trepn-power-profiler, 2018-02-16.

In all other settings, the observed decrease in average goodput was less than 2 Mbit/s, rendering the impact of the power profiling on average goodput negligible.

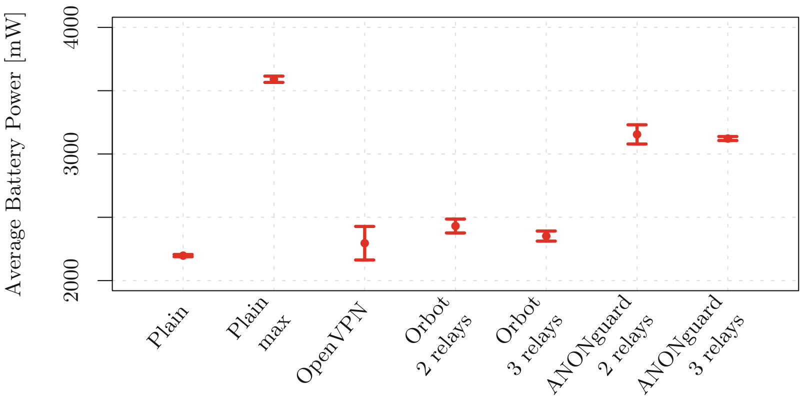

Fig. 6. Mean values for the average energy consumption during a continuous download for different devices and anonymisation solutions.

Results: The results of our measurements are depicted by Fig. 6, where each dot represents the mean value of the average battery power, measured in milliwatt, over 10 runs and the bars denote the confidence intervals for a significance level of 95% based on Student's t-distribution. On maximum screen brightness, we measured an average battery power of 3590.3 ± 24.8 mW during the download if no anonymisation is enabled. On minimal screen brightness, we observed an average battery power of 2197.1 ± 8.6 mW. While OpenVPN and Orbot with 2 relay channels additionally require 98.1 ± 132.4 mW and 234.0 ± 54.8 mW of power on average, our solution currently requires 957.6 ± 75.9 mW of additional power if 2 relays are used. Similar to the results from the previous section, we identified networking-related implementation issues to be the main source of energy consumption. Without cryptography, the ANONguard application still demanded 867.2 ± 47.5 mW of power. If the measurement client contacted the AN.ON mixes directly, 468.7 ± 11.9 mW of power were additionally required without use of cryptography.

Discussion: Ultimately, our measurements indicate that current state of the art solutions like OpenVPN and Orbot only require few additional battery power, suggesting that transparent anonymisation can be implemented without causing prohibitive energy consumption. Our Java-based solution currently consumes significantly more energy than OpenVPN and Orbot, whose anonymisation and VNI handling is implemented natively. However, it remains open if the high energy consumption is inherent in the use of Java or if they can be solved by an improved control and data flow.

9 Conclusion

In this paper we described the design of a low-latency network layer anonymisation solution for mobile devices that uses a virtual network interface to obfuscate network layer information transparently. Our experimental evaluation indicates that current solutions cause a significant loss of achievable goodput but require comparatively low energy. Given the average data rate available for mobile users, we believe the performance and energy consumption impacts are acceptable. Nevertheless we will continue our investigations regarding the performance bottlenecks and further optimise our solution. We also plan to perform further measurement studies to evaluate the performance and latency improvements of the optimisations presented in this paper and to investigate the viability of our solution for latency-critical traffic such as voice over IP or video conferencing.

Acknowledgements. This work was in parts supported by the German Federal Ministry of Education and Research through project AN.ON-Next under Grant No. 16KIS0421.

References

1. Hide My Ass! Blog: Lulzsec fiasco. https://blog.hidemyass.com/lulzsec-fiasco. Accessed 02 Mar 2018
2. Khandelwal, S.: FBI Arrests A Cyberstalker After Shady “No-Logs” VPN Provider Shared User Logs. https://thehackernews.com. Accessed 02 Mar 2018
3. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. Technical report, Naval Research Lab, Washington, D.C. (2004)
4. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: a system for anonymous and unobservable internet access. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 115–129. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44702-4_7
5. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)
6. Wiangsripanawan, R., Susilo, W., Safavi-Naini, R.: Achieving mobility and anonymity in IP-based networks. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 60–79. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76969-9_5
7. Andersson, C., Panchenko, A.: Practical anonymous communication on the mobile internet using Tor. In: SecureComm, pp. 39–48. IEEE (2007)
8. Doswell, S.: Measurement and management of the impact of mobility on low-latency anonymity networks. Ph.D. thesis, Northumbria University (2016)
9. Perkins, C.: IP Mobility Support for IPv4, Revised. RFC 5944, November 2010
10. Google: Connect to Wi-Fi networks - Nexus Help. https://support.google.com/ See “Advanced Network Settings”. Accessed 01 Mar 2018
11. Apple Inc.: Connect to the Internet - iPhone User Guide. https://help.apple.com/. Accessed 01 Mar 2018

12. Android Open Source Project: VpnService—Android Developers. https://developer.android.com. Accessed 02 Mar 2018
13. Apple Inc.: NetworkExtension—Apple Developer Documentation. https://developer.apple.com. Accessed 02 Mar 2018
14. Qualcomm Developer Network: How does Trepn Power measure battery power? https://developer.qualcomm.com. Accessed 02 Mar 2018