Powering Privacy: On the Energy Demand and Feasibility of Anonymity Networks on Smartphones

Daniel Hugenroth
University of Cambridge

Alastair R. Beresford
University of Cambridge

Abstract

Many different anonymity networks have been designed and implemented over the last 20 years. These networks protect communication and metadata through multi-layered encryption and cover traffic. However, there is little research on whether such networks are actually practical on smartphones with limited battery power. This is important as these are the dominant computing devices today. Previous research suggests that cryptographic operations and background radio transmissions are the two main contributors to energy consumption when running such software on mobile devices. We develop and open-source a test setup that measures actual energy consumption, including side-effects that evade simple models. With this setup we explore the costs of cryptography, radio communication, and background scheduling. We find that radio communication dominates overall power consumption, while cryptographic operations (asymmetric and symmetric) are negligible for typical anonymity network workloads. We also investigate the feasibility of running anonymity networks continuously to protect the metadata of all communication. For a 12-hour usage period, a mobile Tor client on a 4G network requires an additional 4 percentage points of battery charge which appears practical and is at least as efficient as the commercial VPN clients that we tested. However, a mix network client that continuously sends cover traffic requires up to 30 percentage points for the same period. Such costs are unlikely to be acceptable to many users.

1 Introduction

Over the last few years end-to-end (E2E) encryption has reached mainstream popularity through mobile messaging applications such as Signal and WhatsApp with billions of users. E2E encryption ensures that only the intended recipients can read a message and that its content is hidden from anyone else, including the messaging service provider. This comes in addition to securing web-based communication through widely adopted standards such as HTTPS.

However, as messages and packets are traveling through third-party infrastructure, most systems still leak metadata such as who is talking to whom, from where, and when. This metadata is easy to capture and interpret for messaging providers, network operators, and global adversaries. This can be dangerous for a whistleblower who reaches out to a journalist, an activist who is researching information online, and diplomats corresponding between embassies.

These risks are even more pronounced when using smartphones. As these devices follow their owner throughout the day, they leak sensitive location data through changing IP addresses and other connection characteristics. Additionally, smartphones roam freely between many different networks such as WiFi and 4G. This exposes the user to many more infrastructure providers that they need to trust. The 2019 Hong Kong protests are one example that shows a large group organizing themselves using mobile devices and requiring access to secure, anonymous communication [1].

Anonymity networks protect metadata and we explore two points in the solution space covering real-time web-based and high-latency message-based communication. We include VPN clients in our evaluation as a baseline benchmark.

Tor [15] is a popular deployment that disguises the real sender and recipient of web-based communication by routing it through multiple onion routers. Its encryption scheme ensures that each hop only learns about its direct predecessor and successor along the chosen path. This makes it hard for an adversary to follow messages through the system even if they have compromised some of the onion routers.

However, a powerful adversary who suspects that two individuals in its influence area communicate with each other can perform traffic analysis to confirm their hypothesis. To do this the adversary captures the traffic patterns at both endpoints A and B. If A is indeed talking to B, an adversary would expect that messages sent by A result in messages received at B shortly afterwards.

Other anonymity network designs, such as mix networks, offer strong metadata privacy for message-based communication to counter such traffic analysis. This is achieved through

traffic shaping where clients pad messages to a common size and inject additional cover packets. An adversary who is observing a participant in such a network will not be able to tell if there is any meaningful communication happening at all. In our work we investigate Loopix [37] which is a practical Mix design and the foundation for the commercial Nym network [14] that runs on thousands of nodes.

However, anonymity networks have not been adequately evaluated for mobile yet, despite smartphones being the dominant computing devices today. Virtually all existing designs have only been tested on stationary computers with reliable Internet connections. Therefore, they overlook the critical challenges of intermittent connectivity and limited energy supply on mobile devices. While the former has been addressed in part by existing protocols, the latter has received little attention. We believe an anonymity network is not practical if it drains the battery too quickly. Therefore, an evaluation of energy consumption is crucial if we are to bring the benefits of anonymity networks to everyone.

While there are existing energy measurement studies, we have not found any that discuss and compare anonymity networks. Those who cover individual operations of interest such as cryptography and radio communication, were done on smartphones that are many generations old. We found that they do not translate to modern smartphones with better hardware and specialized instruction sets.

The current lack of evaluations can be explained by the complexity of mobile energy consumption. First, individual components such as the radio modules contain many internal states. Each data transfer (no matter how small) will incur costly state promotion from *idle* to *connected* where it will stay for a while in anticipation of subsequent transfers. Therefore, the bottom-line energy costs of a data transfer is also influenced by operations before and after. This makes attribution difficult. Second, because these effects are global to the device, the operating system coordinates background work to maximize idle time and tries executing background tasks in batches to make the best use of the radio states. This means that we cannot examine just an app, but need to include the interplay with the surrounding infrastructure. Third, hardware-based power measurements often require extra equipment and expertise which make evaluations appear expensive. We believe that we can make such measurements more accessible.

In this paper we make the following contributions:

1. We develop an open-hardware open-source test setup for ground-truth energy measurements on modern Android devices. Our approach makes it easier for other researchers to perform energy measurements.

2. We provide up-to-date measurements for individual operations (such as data transfer and cryptographic operations) and show that some of them have drastically changed since previous studies were conducted.

3. We are the first to measure and compare the actual energy consumption of anonymity networks on smartphones.

   (a) We show that running Tor continuously in the background on a mobile device requires an additional 4 percentage points (pp) of battery charge during a 12-hour period.
   (b) We show that Tor is at least as energy efficient as two commercial VPN services that we tested.
   (c) We show that anonymity networks with high-frequency Loopix-style cover traffic reduce the overall battery lifetime by more than 20 pp. Only those with very infrequent messages are practical.

4. In the analysis of these measurements we find that running Tor in the background is feasible while low-latency mix networks with cover traffic are not. This points towards the need for new research and designs based on our results.

2 Background and Related Work

We first introduce previous work on energy measurements on smartphones (§2.1). They highlight the practical challenges that motivate our case-studies and demonstrate that we are missing data points from recent device generations. We also give an introduction to anonymity networks (§2.2). Throughout the paper we will use milli-Watt ($1\,\mathrm{mW} = 1\,\mathrm{mV} \cdot A$) as the unit for power and milli-Joule ($1\,\mathrm{mJ} = 1\,\mathrm{mW} \cdot \mathrm{s}$) or milli-Watthours ($1\,\mathrm{mWh} = 3600\,\mathrm{mJ}$) for energy.

2.1 Measuring Energy on Smartphones

Smartphones are ubiquitous and we expect them to do more and more. Therefore, good battery life is key to their utility. In this paper we provide example numbers which assume that smartphones are charged overnight and then used as mobile devices for 12 hours without easy access to charging opportunities. For deciding feasibility we assume that users are able to spend an extra 5% of battery per day. Our results can be easily adjusted to different assumptions if required.

2.1.1 Hardware-Based and Model-Based Approaches

We divide the research field of mobile devices and energy consumption based on measurement approach and scope. The former comprises of hardware-based measurements and model-based approaches that are discussed below. Each of these approaches can be used to examine either individual operations (micro study) or more complex scenarios (macro study).

For *hardware-based measurements* researchers place a power monitor between the power source and the mobile device. They then execute an operation under test while recording the power consumption. Integrating over time (i.e.

calculating the area under the curve) yields the total energy. Hardware-based measurements provide ground-truth results and reflect actual real-world power consumption. Also, hardware-based measurements do not influence the measurements as no additional debugger or tracing software runs on the device. On the downside, they do not attribute the energy consumption to individual components – say measuring only the CPU while ignoring radio communications.

We found that hardware-based setups are often used to capture individual operations. The work by Carrol et al. [6] and Adrito et al. [3] cover many basic operations from display illumination to phone calls. Other work examines specific areas such as Wifi [39], 4G radio communication [18], and machine learning [27]. The GreenMiner [17] project executes pre-recorded app interaction sequences for regression tracking. However, we found that the mentioned existing work concerning the energy consumption of algorithms and radio communications lags multiple generations behind smartphones that are in use today. The recent BatteryLab project [49] focuses on providing remote hardware-based measurements at multiple locations. However, through the more general setup they are less suitable for our research as they provide limited control over the device under test and do not provide the high-resolution synchronization required for analyzing short operations. In terms of equipment, previous work uses either general-purpose power measurement appliances (e.g. Mansoon) or build custom tools using power sensor chips like we do.

On the other hand, there are *model-based measurements* where researchers first create models based on execution traces and then use these to predict the energy costs when running other apps in a second step. These models are simple and effective when the power consumption is dominated by computation as CPU energy consumption is well-documented and their state (e.g. adaptive frequency) can be recorded cheaply. However, radio communications are harder to model as the radio module operates in different states (e.g. connected, idle) with transition latencies depending also on the activity of other apps. In general, models are bound to a specific training device and therefore the model prediction becomes out-dated together with that device. Their utility is further limited as they often need modifications to the apps or operating system (rooting), or use APIs that are no longer present in newer devices. The removal of APIs is often performed by the operating system vendor to reduce side-channel attacks and improve user privacy. For example, access to `/proc/stat` was removed [21] in 2017 citing an attack that exploited interrupt information to recover user input [42].

We found that model-based approaches are typically used to capture complex scenarios that last multiple seconds. PowerTutor [53] models the overall energy consumption of components such as CPU, radio, and display individually. This allows it to achieve high accuracy even for complex scenarios that include GPS and radio usage. However, the calibration devices are now more than 10 years old and many of the required APIs are no longer available. As radio communication is the main driver for many applications, the EnergyBox [51] project from 2014 focuses solely on predicting WiFi and radio energy consumption based on network packet captures. This has been successfully used to examine mobile messaging applications [50] and specific mobile Tor usage [25]. Similar to other model-based approaches, EnergyBox suffers from calibration to older devices and protocol versions.

This literature review is also summarized in Table 6 in Appendix C.

2.1.2 Cryptographic Operations

Our literature review showed that there are no recent micro studies examining cryptographic operations on smartphones. However, such data is important for the design of new protocols as anonymity network implementations make heavy use of encryption, signatures, and key exchanges. The work by Potlapally et al. [38] and Rifa-Pous et al [40] are some of the first to investigate both individual cryptographic operations and protocol executions on mobile devices (PDAs between 2006 and 2011). Montenegro et al. [28] compares the relative energy consumption of different cryptographic libraries on Android. However, since they use the PowerTutor model, their absolute energy predictions refer to Android devices that are more than 10 years old. The most-recent measurement of cryptographic operations that we found is a study of Elliptic Curve Cryptography (ECC) on an ARM-based Internet-of-Things (IoT) platform [29].

2.1.3 Radio Operations

Radio communication is an intrinsically hard area for energy studies due to large numbers of internal states, delayed state transitions, and inter-application effects. When a mobile device starts to communicate via a mobile internet connection (e.g. 4G), the radio module will first promote the system from an idle state (with low stand-by power consumption) to a connected state (that requires more power to maintain). The transition itself costs energy and time, which is why the device will remain in the connected state for a while in anticipation of a response or more data to send. This is referred to as tail latency and its duration depends on the protocol and the mobile provider's network configuration. A delay of multiple seconds is typical. As a result, sending two small packets 30 seconds apart requires more energy than sending one very large packet without interruption.

The work by Huang et al. [18] examines 4G and its power consumption in great detail. Pathak et al. [34] are able to attribute energy costs to individual components, e.g. assigning the resulting radio energy to the background service that caused the transition to the connected state in the first place.

2.1.4 Android

Android is the largest mobile platform with over 3 billion active devices [46]. Since the operating system is open-source, many different manufactures produce and distribute Android smartphones ranging from cheap low-end phones to flagship devices. For the hardware, most manufacturers source components from chipset suppliers such as Arm, MediaTek, Broadcom, or Qualcomm. This allows us to compare smartphones from different manufacturers based on the generation of their internal components.

The operating system plays a key role in managing battery life, as the expectations of end-users and the number of applications grow. One of the most crucial aspects is the coordination of background services and tasks. Android 6 introduced *Doze* and *App-Standby* for this purpose [22]. Doze pauses background execution when the device is idle except for scheduled maintenance windows. By limiting the execution of background tasks to these windows, Android minimizes the number of the state transitions of the radio module. App-Standby further reduces access to background execution for apps by learning which apps are used regularly by the user and placing restrictions on all inactive applications. When applications are found to be rarely used, the system might defer their background tasks by up to 24 hours or restrict background network access [23].

2.2 Anonymity Networks

End-to-end (E2E) encryption protects the content of messages from sender to recipient. However, an adversary can still observe who is communicating with whom and when. Anonymity networks protect this metadata and make it hard for an adversary to learn any information at all. We focus on designs that have smartphone implementations available or that are designed with offline support in mind.

The simplest way to hide some Internet traffic metadata from a local adversary is by using a VPN service. VPN clients capture all outgoing traffic, encrypt it, and send it via the VPN operator's server. This technique is employed both by companies which operate their own servers and individuals who subscribe to commercial VPN providers. Individuals might use a VPN service to protect their traffic on insecure WiFi (e.g., in a public cafe) or to access streaming services in other countries. However, it is easy for a rogue VPN provider to link the traffic from its servers with individuals. Therefore, VPN networks provide very limited anonymity against dedicated adversaries. In our evaluation we measure two commercial VPN services targeted at regular mobile users: ExpressVPN and Proton VPN. Both have more than 10 million downloads on the Google Play Store.

Decentralized anonymity networks avoid the design weakness of trusting a single operator. Among these, Tor [15] is the most popular and practical anonymity network design deployed today. Clients hide their relationship with the other party by sending their data via three onion routers. Each onion router can only decrypt the outer-most layer of the message to learn the address of the next hop and the inner (encrypted) packet to forward. An attacker would need to compromise all onion routers along the path to learn the full path between the sender and recipient. We focus only on running end-user Tor clients and not onion routers.

However, Tor is susceptible to traffic analysis [13, 30]. An adversary who suspects that two specific individuals are communicating with each other can record their encrypted traffic and then correlate timing patterns. If the adversary repeatedly observes that, when A sends a packet, a packet arrive shortly after at B, then the adversary might conclude that A and B are likely communicating. Making this observation repeatedly increases the adversary's confidence. Tor can make such attacks more difficult by sending so called cover traffic on top of the user-generated traffic [35]. While it does not provide full protection, it increases the cost for the adversary [24].

For our evaluation we use the Orbot [44] app released by the Guardian Project and endorsed by the Tor Project. It is a popular app with more then 10 million downloads from the Android Play Store. Besides providing anonymity, it can also be used for censorship circumvention. The Orbot app includes the official Tor client and emulates a VPN client on the device. This allows the user to choose to send either all traffic or that of selected apps through Tor. We use Orbot for our case study (§4.5). Alternatives, such as the official Tor Browser app, only protect traffic generated by the bundled browser and not for other apps [45].

Other anonymity network designs can achieve unobservable communication [36] where an adversary is unable to tell whether any communication is happening at all. This is typically done by using traffic shaping. With this technique all messages are padded to a fixed size before being encrypted and sent at pre-determined times. Those times can be at fixed intervals or drawn from random distributions – the key requirement is that they are chosen independently of whether there is real data to send or not. If there is no real traffic (e.g. the user is idle), an empty message is made-up and sent instead. If multiple real messages arrive at once, they are queued and sent one after the other.

Mix networks [5, 7, 8, 26, 37, 41] use this technique and route messages independently through multiple mix nodes — whereas in Tor all messages are sent via the same route. Each mix node delays messages independently which makes traffic analysis even more difficult. Both the message timings at the clients and the delays at the Mix Nodes are drawn from an exponential distribution resulting in a Poisson process. Loopix [37] and Groove [5] are mix networks designed with mobile devices in mind. Both account for mobile clients that can be offline through inboxes at provider nodes that can store incoming messages until the client comes online again. We also chose Loopix in our evaluation, because its design is

Name	Ref.	Year	Anonymity	Bandwidth	Latency	CPU	Mobile Devices
Dissent	[52]	2012	◐	●	●	●	○[1]
Vuvuzela	[48]	2015	●	●	●	●	○
Hornet	[9]	2015	●	◐	●	◐	○
Riposte	[11]	2015	◐	●	◐	●	○
cMix	[8]	2017	●	○	●	●	○
Loopix	[37]	2017	●	●	●	●	◐[2]
Groove	[5]	2022	●	●	●	●	◐

Table 1: Summary of evaluation metrics used in recent anonymity network papers (● = thoroughly covered, ◐ = covered, ○ = not covered); [1]mentioned as important future work; [2]offline support.

currently practically deployed as part of the Nym network [14] with hundreds of mix nodes [33].

We have reviewed the evaluation methods used in widely-cited and claimed to be practical anonymity network designs. The results are summarized in Table 1. We limited ourselves to recent publications which we would expect to cover mobile devices. Most work focuses on the bandwidth and achievable performance in terms of throughput and latency. The required computational cost (i.e. CPU) is often only considered to the extend necessary to rule it out as a potential bottleneck. Private information retrieval (PIR) based networks [2, 10, 11, 16] are the exception in this regard as they typically require costly computation from both the clients and servers. Most publications also quantify the achieved anonymity in terms of either anonymity set size or entropy.

Overall, there is little consideration for the practicalities of running these networks on mobile devices. Notable exceptions are Groove [5] which measures its power consumption, Loopix [37] which provides support for offline clients, Dissent [52] which flags this as important future work, and Hydra [41] which highlights the difficulties of precisely scheduling background execution on Android (see §2.1.4). Groove is the only one to provide measurements of its energy consumption on a smartphone. Unfortunately, their methodology limits the comparability and reproducibility of their results. Their setup measures the charging rate of the device instead of the power drawn by its components. In addition, apps and OS might undertake more work when they detect the device is charging. Their measured idle power consumption of 310 mW makes the discrepancy clear as it is much higher than a typical smartphone idle consumption of <100 mW. Also, their sampling frequency (once per second) is too low to capture peaks from cryptographic operations and transmissions.

3 Measuring Energy Consumption

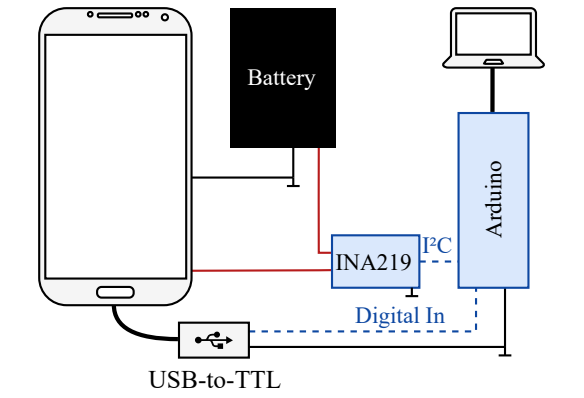
We use a hardware-based approach for our measurements for three main reasons. First, we have not found any model for recent smartphones that covers both CPU and radio communication. Second, in our case studies we are interested in real-world battery life which must include the interference with the operating system scheduler, all utilized components, and side-effects. Third, we are interested in overall execution profiles that include the lowest standby states. Running software-based debuggers or tracers in the background would prevent the smartphone from entering these.

In our setup we install a Texas Instruments INA219 [43] power sensor between the smartphone and its battery. This sensor records 2,000 power measurements per second with 1% accuracy using a shunt resistor. An Arduino polls these samples via an I^2C bus and forwards them via USB to a computer that records timestamps and power values into a power measurement CSV file.

For our multi-second macro studies this setup is already practical as the operator can start and stop the measurement manually. However, for micro studies where individual operations only run for a few milliseconds we automate this process by creating the custom app *EnergyRunner* that executes the individual operations and records timestamps in an *execution log* on the device. We synchronize the time on the smartphone and the power measurement logs using an USB-to-Serial dongle (FTDI FT231X) that is connected to the smartphone's USB port. Its ground is connected to the common ground of the Arduino and one serial control line (we use RTS) is connected to a digital input port of the Arduino. The Arduino reports changes to the digital input via the same USB connection that is used for the power measurement samples. The synchronization sequence is executed before the experiment so that we can later correct for differences in *clock offset*. We found that differences in *clock speed* are negligible (less than 0.001ms per hour). The USB dongle is disconnected before the measurement of the actual operations start to ensure that its own power consumption does not influence our measurements.

Our choice of smartphones and hardware allows the tests to be done without permanent hardware modification and requires minimal technical skills. This is especially true for our smartphone where the battery can be removed without tools and placed into our 3D printed battery holder. The full hardware setup is illustrated in Figure 1.

We now describe the protocol for running an experiment with our setup. First, our instrumentation app EnergyRunner loads a scenario file which describes the order and parameters of the operations that we want to execute. It automatically adds the synchronization sequence at the beginning of the execution schedule. We first start the recording software on the PC and then start the execution on the smartphone. During execution, EnergyRunner records the timestamps of operation



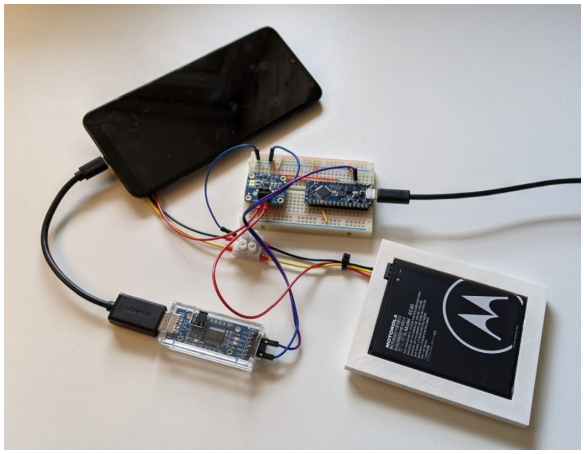


Figure 1: Schematic and photograph of our hardware setup.

start and end. Pauses between individual operations are described in the scenario file as well. After the entire execution schedule has finished, the execution log with the timestamps is saved and uploaded. Finally, our processing scripts will read both the execution log from the Android device and the power measurement CSV file. The processing code identifies the synchronization patterns to adjust for clock differences and then extracts the power measurements for each individual operation based on the timestamps in the execution log.

We prepare the device under test by uninstalling and deactivating all apps that can cause background activity such as Google Play Services. With the start of the execution of the scenario files the display and all radio connections are turned off unless needed by the operations under test.

Using this setup we can test individual operations semi-automatically, efficiently, and accurately. As we hope that such energy measurements become more common in papers that introduce (mobile) protocols, we put a lot of effort into making sure that this setup can be easily replicated by other researchers. All of our software for executing operations, logging the data, and analyzing the results is shared as open-source. It also includes an interactive logging tool that shows incoming data in a live plot. Likewise, all our hardware specifications, 3D printing files, and assembly instructions are part of the repository. All is available at https://github.com/lambdapioneer/powering-privacy under an MIT license. The hardware components are widely available, cheap, and easy to assemble.

All studies are performed using the setup described in this section and we use a Motorola Moto E6 Plus (Released September 2019) smartphone running Android 9. It comes with an MT6762 Helio P22 chipset, a 2.0 GHz Cortex-A53 CPU, and 2 GiB RAM. We have updated the smartphone to the most recent OS update and have uninstalled and disabled all other applications.

4 Evaluation

We start our evaluation by studying individual cryptographic operations (§4.1), background scheduling (§4.2), and radio transmission (§4.3). These are the important building blocks for anonymity network protocols and help interpreting the results of the macro studies.

In the first macro study we investigate the impact of Orbot [44]. We are interested in the effect on battery life for different configuration and scenarios. The second macro study measures a mix network with cover traffic. For this we implement a mobile client for Loopix and evaluate different sets of parameters. In total, our evaluations cover more than 100 hours of recorded power samples. The raw data and evaluation scripts are available from our repository and can be used to reproduce the results shown in this paper.

For short operations with a clearly defined start and end we provide the energy consumed in milli-Joule ($1\,\text{mJ} = 1\,\text{mW}\cdot\text{s}$). Battery capacity is provided in milli-Watthours ($1\,\text{mWh} = 3600\,\text{mJ}$). For continuous applications (e.g. running an anonymity network in the background) we provide the average power in milli-Watt ($1\,\text{mW} = 1\,\text{mV}\cdot A$). Where helpful, we also translate the average power into how many percentage points of the total battery capacity this application will consume per hour (pp/h). For the remainder of the paper we use an effective battery capacity of 8000 mWh for our calculations. This is a typical battery size found in a range of mid-to-high-end smartphones [47].

4.1 Micro Study: Cryptographic Operations

In this micro study we evaluate to total energy (mJ) impact of individual cryptographic operations. All results are summarized in Table 2.

Asymmetric operations

Asymmetric cryptography is frequently used in anonymity network protocols for signatures and encryption of the payload. At the same time it has a reputation for being computationally intensive. Many anonymity networks use a message

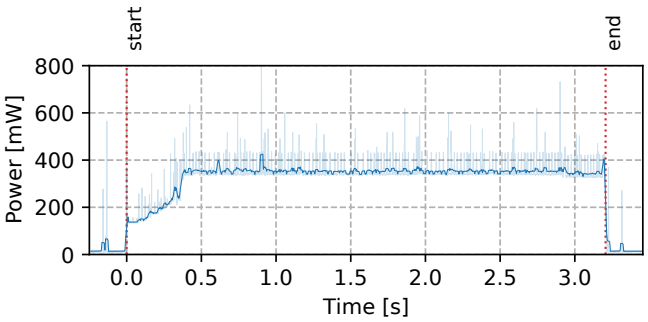


Figure 2: Power trace of generating a 2048-bit RSA key pair. The power rises between 0.0 s and 0.4 s as the system increases the core frequencies. The light blue line shows the raw measurements while the dark line is a rolling average.

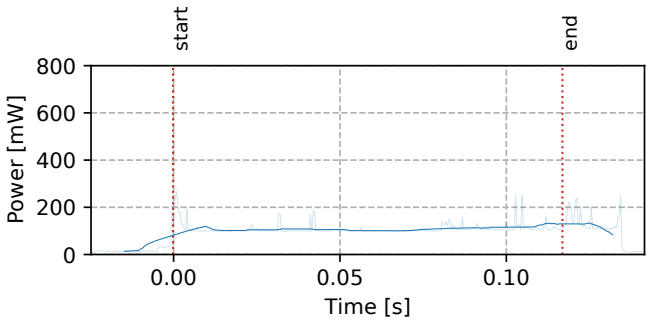


Figure 3: Power trace of a single Sphinx packet creation execution including JNI overhead.

based architecture (as opposed to a channel based one) where typically no key agreement between communication partners occurs. This means that every message that is sent or received involves asymmetric operations. Also, in multi-hop based architectures encrypting messages for each hop along the path further increases the number of asymmetric operations.

RSA is an established asymmetric cryptographic scheme that is still in common use. Its key generation algorithm involves finding prime numbers which can lead to long runtimes (see Figure 2) and high variance due to non-determinism. Key generation for 4096-bit RSA can require up to 2800 mJ which translates to just 14,500 executions with one battery charge. However, once a key is available, sign and verify operations are cheap. As RSA implementations use a high exponent for the private key and a small one for the public key, verify operations are much faster. If the exponent sizes are swapped (e.g. DSA), signing become cheaper.

In the last 20 years RSA has been slowly phased out in favor of Elliptic Curve (EC) cryptography. Its smaller key sizes generally allow fast execution, compact representation, and it does not require expensive prime-number search for key generation. We find that all operations are (negligibly) cheap and more efficient than those of equivalent RSA key sizes (256-bit EC is considered equivalent to 3072-bit RSA). We tested the four different EC curves commonly available on all Android versions through the built-in Android OpenSSL provider [19]. The `prime256v1`/`secp256r1` cipher is notably faster than the others.

It is possible to build a packet format for mix networks using these EC operations. However, actual implementations use specialized cryptographic constructions such as Sphinx [12] which allow for more compact packets and additional security properties. We evaluate a modern Rust implementation of Sphinx [32] (git commit: `c494250`) that is used in the commercial mix network Nym [14]. Nym is based on Loopix [37] which also uses the Sphinx packet format. For our experiments we add a JNI binding to the library to allow calling it in our Android app. Since the JNI boundary adds a non-negligible overhead for such small operations, we add an iteration parameter so we can execute multiple rounds without leaving the native code. Each round we call `SphinxPacket::new()` which internally uses `Curve25519` primitives. We found that this Sphinx implementation is efficient (see Figure 3) and comparable to a few EC operations. The JNI overhead is up to 15% based on the difference between the configurations with 1× and 100× iterations.

Our results show a significant reduction in energy costs compared to two other studies that use a hardware-based approach. Rifà-Pous et al. measured RSA and EC operations on PDA devices in 2010 [40], and Mössinger et al. measured EC operations a ARM-based development board in 2016 [40]. Compared to our results, the energy costs of the RSA operations on the PDAs are higher by factor ×3 (Verify RSA-1024) to ×20 (Sign RSA-2048). The costs for EC-224 operations on the PDAs are higher by factor ×30. The ARM-based board using the optimized MicroECC library requires around ×20 times more energy than our test device. More data points are available in Table 5 in Appendix B.

Other operations

We found that all standard hash operations are negligibly cheap. This is due to the low computational complexity and the wide availability of specialized CPU instructions such as `SHA256H2` and `AESE` on Arm64 [4, p.1556]. For completeness we also measured the energy required to hash a 16 KiB byte array with SHA-256/512 resulting in energy costs of 0.26 mJ and 0.27 mJ respectively.

4.2 Micro Study: Scheduling

In this micro study we quantify mechanisms for executing code while the phone is not actively used. These allow applications to perform message synchronization and background computations. Anonymity networks use them for sending cover traffic during idle mode. This is critical for hiding whether a user is currently communicating or not. We focus

Operation	Energy [mJ]	StdDev
Gen RSA-4096	2898.43	2042.77
Sign RSA-4096	26.73	4.68
Verify RSA-4096	0.75	0.10
Gen EC-256	0.51	0.05
Sign EC-256	0.83	0.18
Verify EC-256	1.54	0.08
Sphinx (1×)	9.66	0.31
Sphinx (100×)	842.44	11.40

Table 2: Average energy consumption of different asymmetric cryptography operations.

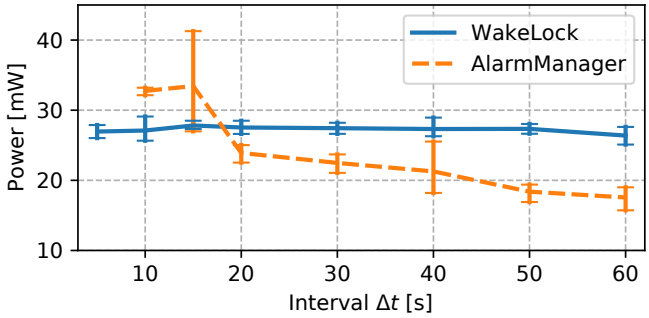


Figure 4: Average power consumption when using the *ForegroundService with WakeLock* and *AlarmManager* strategies.

on the two main strategies for Android: *ForegroundService with WakeLock* and *AlarmManager*. Devices running iOS have similar, albeit more restrictive, mechanisms [20].

For our purposes we discuss these mechanisms as they were intended by the operating system design. Many smartphone vendors make modifications and add restrictions to achieve higher battery life which can interfere with the proper operation of apps [31]. We verified that the device we use in our studies follows the specified behaviors.

The *ForegroundService with WakeLock* strategy holds a lock that prevents the phone from entering full idle mode. As this affects background power consumption, the operating system requires the app to show a notification to the user by running as a ForegroundService. This approach provides the greatest flexibility and accuracy, as pauses can be implemented as simple `Thread#sleep` calls. Typically, this mode of operation is used by music apps and GPS navigation. However, even without active computation, the WakeLock causes higher energy consumption than regular idle mode.

The *AlarmManager* strategy allows the phone to go into full idle mode and registers our intended execution with an alarm service. The alarm service then wakes up the device and starts our application. While this allows the phone to enter full idle mode between executions, every wake-up comes with overhead as the system restores state and (re-)delivers the invocation arguments to our application. The AlarmManager's intended use are events that happen only a few times per hour. We found that the lowest reliable inter-execution pause is 10 seconds and that the execution times are imprecise. We compensate for the latter by measuring the time between the scheduled and the actual execution, and then applying this delta when scheduling the next alarm. However, this compensation is imperfect as the effective delays vary.

We use Δt to denote the duration between the start times of two consecutive operations. We expect that using WakeLocks is more efficient for smaller Δt. While keeping the CPU awake generally raises power consumption, WakeLocks do not cause extra overhead for every execution. On the other hand, the AlarmManager should perform better for larger Δt as it allows the phone to reduce power consumption during longer pauses.

In our experiment we trigger regular execution for various intervals Δt with both the WakeLock and the AlarmManager approach. Each experiment runs for 5 minutes and is repeated 5 times. The results are shown in Figure 4. As expected, the WakeLock approach incurs constant cost ($\approx 27\,\text{mW}$) regardless of the chosen interval. The AlarmManager approach is more efficient for $\Delta t > 20\,\text{s}$. Generally, it does not differ more than $\approx 10\,\text{mW}$ from the WakeLock approach in either direction. We have included annotated power traces for both approaches in Appendix A. These highlight the additional power consumption that happens before and after execution of our code for the WakeLock approach.

4.3 Micro Study: Radio Operations

We evaluate radio operations such as sending and receiving data via WiFi and mobile network to measure the impact of connection type, payload size, and schedule. The results from this study guide us in parameterizing and evaluating a mix network client with cover traffic in §4.6.

As mentioned in §2.1.3, radio communication, and in particular mobile networks, are complex due to their internal state machine and tail latencies. This means that their behavior applies globally to the smartphone and not for each app independently. Hence, we cannot consider transfers individually, but need to include the effects of previous and concurrent transfers. Figure 7 shows an annotated measurement of a TCP data transfer via 4G. While WiFi has negligible tail latencies, we found that it is more susceptible to noise from other devices, as the smartphone is woken up regularly to process incoming broadcast packets.

In this study we connect to a 4G mobile network using a prepaid SIM card from the UK provider GiffGaff which uses the O2 network. We have decided not to include results for 3G since 4G is widely supported and more popular. Before each experiment we verified that the mobile phone has good reception. For the WiFi measurements, we setup a private access point that is secured using WPA2-PSK which is a typical setup for many consumer routers.

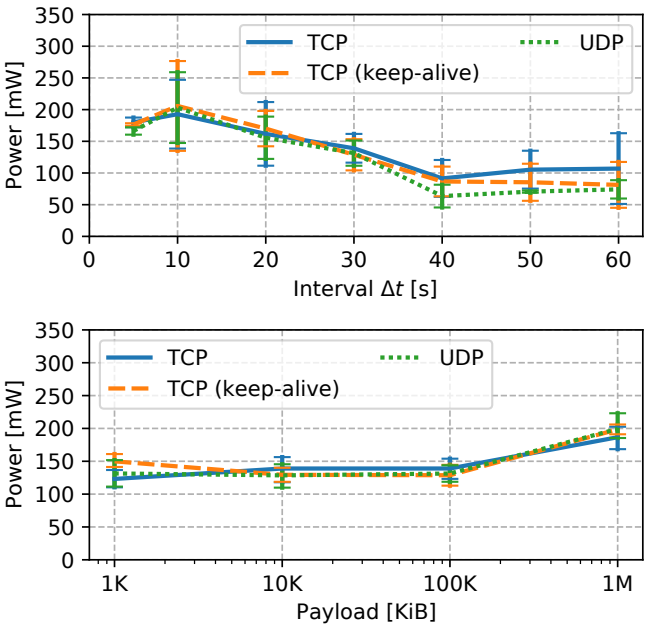


Figure 5: Top: average power consumption when sending and receiving 100 KiB with increasing interval times over 4G. Bottom: average power consumption when sending and receiving increasing payload sizes at a fixed interval of $\Delta t = 30$ seconds.

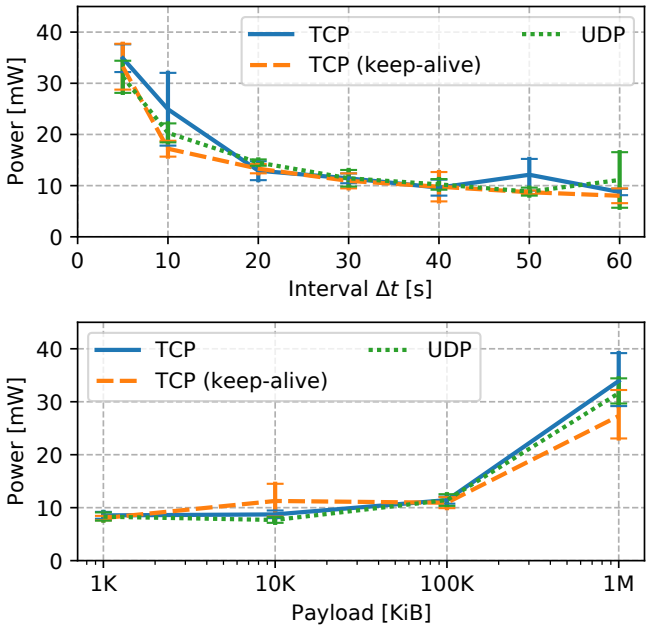


Figure 6: Same as in Figure 5, but for WiFi.

We execute data transfers using the EnergyRunner app. The app uses the WakeLock method to control the scheduling of background operations. We chose the WakeLock over AlarmManager, as its average power consumption is constant regardless of the chosen interval length. This allows us to subtract it from the measured data in order to separated out the costs for radio communication. The network operations connect to a custom server that we run on a virtual machine which is located in a nearby city (ping $< 100\,ms$). In our protocol the message consists of a 16 byte secret token (this is to reduce abuse by third-parties), a 4 byte client length field, and a 4 byte server length field. The server continues reading until it has consumed all data specified by the client length field and then responds with data as specified by the server length field. We always set both to the same size.

We test three protocols: TCP, TCP (keep-alive), and UDP. In *TCP* mode a new connection is established for each individual transfer. In *TCP (keep-alive)* mode the app maintains a global socket connection that is used for subsequent transfers. The latter can reduce the total number of round-trips by avoiding the handshakes for each separate transfer.

We evaluate both different intervals and different payload lengths. For the former we increase the interval from 5 seconds to 60 seconds with a fixed payload size of 100 KiB. The different payload lengths range from 1 KiB to 1 MiB with a fixed interval of 30 seconds. Each configuration is tested via 4G and WiFi for 5 minutes each. We repeat all experiments 5 times and randomize their order to account for noise.

Our results for 4G are summarized in Figure 5. Generally, for increasing interval times, the average power consumption drops. Notably, there is an exception for $\Delta t \approx 10\,s$. With this configuration the device reconnects right after the tail latency has expired, hence maximizing power consumption. We found that payload sizes up to 100 KiB had little impact on the average power consumption. Our WiFi results are shown in Figure 6. We found that WiFi has much lower power consumption compared to 4G. Without the connection establishment and tail latency, the effect of increasing intervals shows more directly. Similarly, the payload size has a more direct impact.

4.4 Macro Study: VPN

In this section we examine the question of how much VPN clients impact battery life. For this we examine two popular, commercial services ExpressVPN (version 10.89) and Proton VPN (version 4.6.12) which are marketed for personal use on mobile devices. We randomly blind them as VPN_A and VPN_B. We created paid-for subscription accounts with both providers and then installed their most-recent apps on our test device. Where possible we use the default configuration and connect to an end-point in New York. For Proton VPN we changed the protocol from *Wireguard UDP* to *Wireguard TCP*, as the former was less reliable in our setup.

Our measurement setup captures the total device power consumption while the VPN service is running. This ensures that we include all direct and indirect effects that affect battery life. We run an *idle scenario* where no user action is simulated and

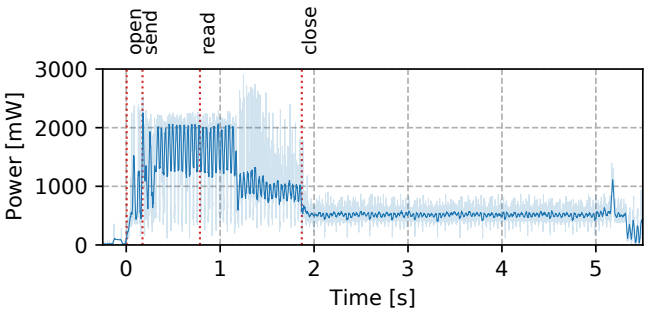


Figure 7: Power trace of sending and receiving 1 MiB over TCP on a 4G mobile network. When comparing with other the power traces note the larger range on the Y-Axis.

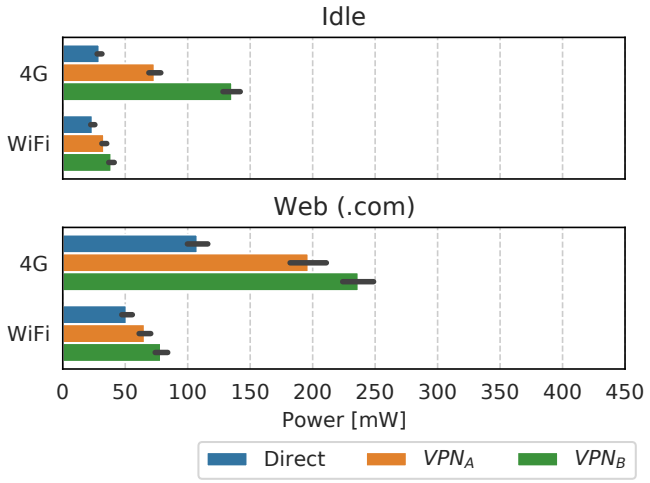


Figure 8: Relative power usage of the tested VPNs compared to direct network connections.

a *web browsing scenario* where an instrumented web browser loads the start page of the New York Times every 60 seconds using the `AlarmManager` scheduling strategy. The news site was chosen as it has a typical size ($\approx 3\,MiB$) and requires multiple connections to different domains. The AlarmManager strategy was chosen to allow return to idle. Our implementation holds a temporary `WakeLock` while loading the page until the Chromium-based `WebView` signals completion.

We run all combinations of activities (idle and web browsing), radio (4G and WiFi), and network configuration (direct, VPN_A, and VPN_B). Each run lasts 10 minutes and similarly to the previous studies we deactivate all other apps and keep the screen off.

Our results are summarized in Figure 8. First we look at the baseline measurements without any VPN service (direct). The idle scenario shows that smartphones can achieve very low power operations when there is no background activity. Notably, the active loading of the website has a strong effect for 4G, but not on WiFi. This is likely due to the aforementioned connection state changes and tail latency.

Both VPN providers have similar overhead. Where they differ we pick the lower power consumption, as we are interested in a competitive baseline to compare Tor and Loopix against. During the idle scenario, a VPN adds around 40 mW or 0.5 percentage points per hour (pp/h) on 4G. This is less pronounced on WiFi due to negligible tail latencies: 8 mW (0.1 pp/h). For web browsing, the overhead increases to 80 mW (1.0 pp/h) on 4G and 20 mW (0.2 pp/h) on WiFi.

We expected a smaller overhead from the VPN apps – especially when the device is idle. Inspection of the acquired power traces shows that while the clients are running, the device does not reach low-power mode, but instead regularly wakes up to send and receive data. This is surprising, as the tested Tor client (§4.5) is able to run with slightly less power overhead. We have cross-checked the results with an OpenVPN client and a self-hosted server which led to similar measurements. We suggest future work to explore the VPN client implementation and configuration space in full detail.

4.5 Macro Study: Tor

In this section we examine the question of how much Tor impacts battery life of mobile devices. For this we examine Orbot [44], which uses the official Tor client under the hood. We downloaded version `16.6.0-RC-4` from the official repository and added an option to force off connection padding (see below). The app is compiled in release mode and then installed on the test device. For our evaluation we run Orbot in VPN mode which emulates a VPN client on the device and ensures that all communication is routed through Tor. This is different to the behavior of the Tor Browser which only protects the communication of the bundled browser. We use the same measurement setup as for the VPN services in §4.4.

Tor supports rudimentary cover traffic through connection and circuit padding [35]. Connection padding affects the connection to the first hop (the Guard node). When active each payload packet causes both ends (i.e. the client and the Guard node) to sample a timeout between 1.5 and 9.5 seconds. If no other payload packet is sent before the timeout expires at either end, a single padding cell packet is sent and the timeouts are reset. To reduce the overall overhead, the client may negotiate a reduced mode where the Guard node does not send padding cells and the client samples a timeout between 9.0 and 14.0 seconds. A client may also completely disable connection padding. We test all three variations (full, reduced, disabled) of connection padding as they have a large impact on radio communication.

Circuit padding is independent of connection padding and aims to obfuscate the setup phase of onion circuits. The client will send obfuscated packet sequences so that different circuit types result in similar looking packet exchanges. We verified that circuit padding has negligible impact regardless of the chosen connection padding. Hence, we do not include it as a parameter for our experiments.

As in §4.4, we run all combinations of activities (idle and web browsing), radio (4G and WiFi), and network configuration (direct and the various Tor connection padding modes). We exclude the initial connection phase to the Tor network as this a one-time cost. The Tor Consensus document ($\approx 600\,KiB$ compressed) needs to be updated every three hours. We argue that it is negligible small compared to other web traffic.

Our results are summarized in Figure 9. For discussion of the *direct* connection without any services running, see §4.4. Using Tor without any padding has little effect on the idle scenarios. From our measurements we calculate an average power consumption of 59 mW or 0.7 percentage points (pp) per hour (WiFi: 0.3 pp/h). For 4G the power consumption increases by <30 mW compared to not using Tor whereas the impact is negligible on WiFi. However, for the web browsing scenario, enabling Tor increases the average costs by around 150 mW (WiFi: 60 mW). This is mostly due the lower bandwidth and high latency which both increase the time website's loading time and hence the time the radio module is active.

Enabling full connection padding in Tor has a significant effect on battery life. On 4G the average idle power consumption increases to 2.9 pp/h (WiFi: 1.1 pp/h). The relative factor for 4G (7×) is higher than that of WiFi (3.5×). The difference can be explained by the costs on 4G for the connection state changes. With reduced padding the average power costs are lower than the average increase of the interval would suggest. We found that this is because the longer gaps allow the entire system to enter a low-power state from which it often does not wake up again itself. Examination of the source code showed that Orbot itself does not use `WakeLocks` or `AlarmManager` – so it does not prevent the system from entering idle mode. Note that our web browsing scenario wakes up the entire system including Tor which then stays active for a while. This explains why the impact of Tor is larger in the Web scenarios.

Tor can also be used to access Onion Services by using their .onion address. Through a directory look-up and an anonymous rendezvous point, the client establishes an anonymous connection with the service without learning its server IP or location. The New York Times offers such an Onion Service. Overall we observe a slightly increased power consumption compared to opening the regular homepage via Tor, which can be explained by the multi-round connection setup and the longer path to connect to an onion service.

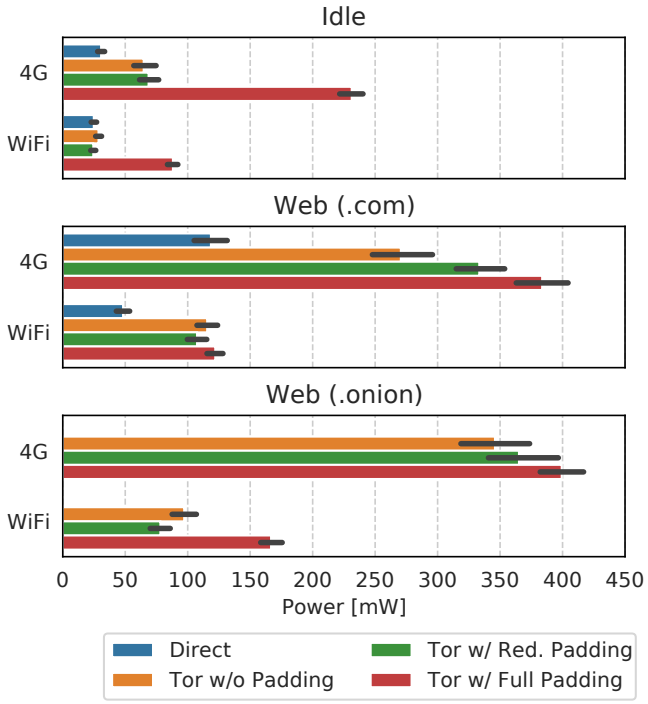


Figure 9: Relative power usage of different Tor modes compared to direct network connections.

4.6 Macro Study: Mix Network

Finally, we evaluate mix networks operations based on the Loopix [37] design. We chose it because of its integrated support for devices that are temporarily offline and note that it is successfully deployed in a commercial mix network called Nym [14]. In Loopix the provider node manages access to the network and maintains an inbox of messages for the client. This allows the client to retrieve messages later when it was offline. Hence, all connections of the client go through one provider node which simplifies our setup as we can ignore the rest of the network. Loopix clients use traffic shaping to hide whether actual communication is happening or not. Inter-packet delays are drawn from an exponential distribution with parameter λ (messages per second). If there is at least one message in the outgoing payload queue, the oldest one is sent (FIFO). If there is no payload message, a cover message is created and sent instead. This means that in our evaluation we can ignore the presence of actual payload messages, as they neither influence the number of cryptographic operations nor have an influence on the sent traffic schedule and bandwidth. Hence, all cryptographic operations and radio transmission solely depend on the message rate λ and message size p.

Our evaluation uses the *WakeLock* approach for background scheduling. For each round we execute Sphinx once for the outgoing packet and transmit a message of size p (the encrypted packet) via UDP to a provider node. The provider replies with p bytes (the inbox content). Afterwards we draw a pause from the exponential distribution with parameter λ and wait for the remaining time until the start of the next operation. Each scenario is executed for 20 minutes on both 4G and WiFi. For our parameter choice of λ and p we first identify practical limits. On our test device a Sphinx operation and sending a UDP packet takes around 20 ms. Using a speed test we measured that our 4G connection provides up to 2 Mbits/s upload (WiFi: 50 Mbit/s) and exclude configurations exceeding these limits. We pick 2 KiB as our smallest

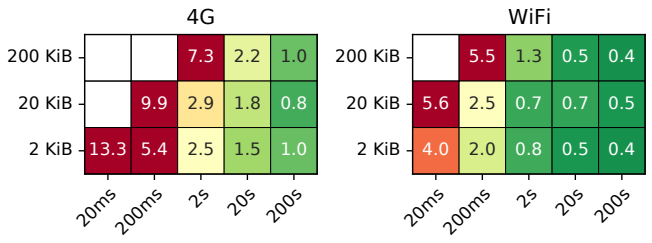


Figure 10: Measured energy consumption of a Loopix-style anonymity network for given mean message intervals ($1/\lambda$) and packet size (p). The color scale ranges from ● 0 pp/h to ● 5 pp/h. Blank squares indicate excluded configurations.

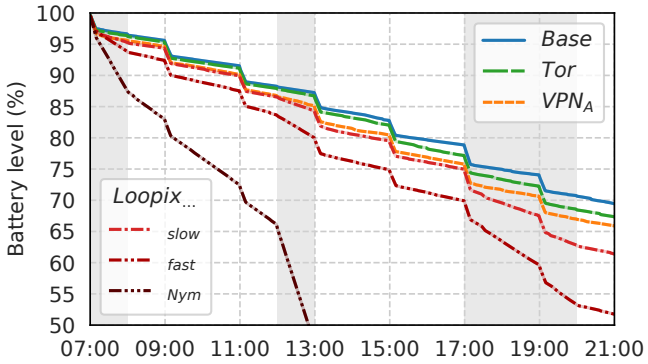


Figure 11: Measured energy consumption as estimated battery levels during our 14 hour daily driver scenario. A white background indicates connection via WiFi (gray: 4G).

packet size. This is also the size used by Nym. For $\lambda = 20$ ms the theoretical throughput is 100 KiB/s (40 % of the available bandwidth). In the Sphinx implementation each packet has a constant overhead of 365 bytes resulting in an application level goodput of 82 KiB/s. Larger packet sizes improve the goodput to throughput ratio.

Our results are shown in Figure 10. By using the same factors for both parameter scales, data points on diagonals (up and to the right) share the same bandwidth. For $\frac{1}{\lambda} = 20$ ms on 4G both CPU and network are consistently active and deplete the entire battery in 7.5 hours. The results also show that for the same bandwidth, larger intervals lead to significant energy savings. For example, with 10 KiB/s of bandwidth, the energy used decreases from 5.4 pp/h ($p = 2$ KiB, $\frac{1}{\lambda}$=200ms) to 2.2 pp/h ($p = 200$ KiB, $\frac{1}{\lambda} = 20$ s) in the 4G case and from 2.0 pp/h to 0.5 pp/h for the same parameters in the Wifi case. However, this comes at the cost of increased latency.

4.7 Macro Study: Daily Driver

The results from our macro studies suggest that some configurations can be run continuously during normal daily usage. To validate these conclusions, we devise a "Daily Driver" scenario that mimicks typical smartphone usage while running anonymity networks. Assuming an informed user, who prefers the most energy-efficient configuration, we choose VPN_A and *Tor without padding*. For the mix network we pick two parameters from the Loopix paper as $Loopix_{fast}$ ($\frac{1}{\lambda} = 2$ s) and $Loopix_{slow}$ ($\frac{1}{\lambda} = 20$ s). As the paper does not mention concrete packet sizes, we choose $p = 20$ KiB which can handle long text messages. We also include the default Nym configuration ($\frac{1}{\lambda} = 20$ ms, $p = 2$ KiB) as $Loopix_{Nym}$. All are compared to a *Base* configuration without any anonymity network client.

Our experiments are performed continuously for 14 hours simulating a typical day from 7 am to 9 pm. The smartphone has been reset to factory state, fully charged, and the following applications are installed: Google Mail, Signal, YouTube. The device is connected to WiFi most of the day, except for 7-8am, 12-1pm, and 5-8pm when it then connects via 4G. We send an email to the Gmail app every hour and a text message using Signal every 15 minutes. Both wake up the device to show a notification. In addition, we play a 10-minute YouTube video every two hours starting at 7am to simulate longer Internet sessions with multiple HTTP requests. The display brightness is set to 75%.

Our results are shown in Figure 11. Unsurprisingly, most energy is consumed when the screen is active (around 1,000 mW). Push-notifications appear as small drops every 15 minutes. Overall, the findings agree with our previous experiments. In this specific scenario the overall energy overhead for *Tor* and VPN_A are 0.2 pp/h and 0.3 pp/h respectively. Hence, both appear practical. The change in the gradients for Loopix make the power consumption differences between WiFi and 4G clear (see e.g. $Loopix_{fast}$ around 12 noon). The high-latency $Loopix_{slow}$ configuration has an average overhead 0.6 pp/h ($Loopix_{fast}$: 1.3 pp/h) which is almost practical when there is WiFi for most of the day. The $Loopix_{Nym}$ parameters drain the battery completely after 12 hours.

4.8 Discussion on Feasibility

We discuss the feasibility of VPNs, Tor, and mix networks with cover traffic based on our assumptions from §2.1: smartphones are used without charging for a 12-hour period and using less than an extra 5 percentage points of battery during this time is acceptable. Our results can be easily adjusted if required. We benchmark the anonymity networks against an idle phone with active network connection which consumes around 0.4 percentage points per hour.

We evaluate whether it is feasible to run Tor without padding in the background by considering the idle scenario on 4G. This gives us a lower bound of the actual costs. Note that active usage and background communication will experience additional overhead due to higher latency and lower bandwidth. For a 12-hour usage period running Tor without connec-

tion padding on 4G requires an extra 3.6 pp[1] (WiFi <0.1 pp). This is less overhead than from the tested VPN clients (§4.4). We conclude that Tor without connection padding is feasible on modern smartphones and can be run continuously without large drawbacks. However, full connection padding has a large impact. This configuration increases energy costs by 30.0 pp on 4G and 9.6 pp on WiFi which is no longer practical.

We evaluate the Loopix-style mix network using the same methodology. In contrast to Tor, active usage does not increase the overall energy overhead of the mix network client as the scheduling and size of packets does not change. We consider a packet size of 20 KiB for both a medium-latency ($\frac{1}{\lambda} = 2$ s) and a high-latency ($\frac{1}{\lambda} = 20$ s) configuration. A 2 second latency would be acceptable for text-based chats and our evaluation shows that it requires an additional 30.0 pp on 4G for a 12-hour period. The high-latency configuration fares better with 16.8 pp for 12 hours. While the numbers are acceptable on WiFi (both 4.8 pp), using local networks can not fully compensate for the high energy costs of 4G networks, as we expect smartphone users to use their devices in many different locations. Only very high-latency parameters (e.g. $\frac{1}{\lambda} = 200$ s) would have a small enough overhead (4G: 6 pp, WiFi: 2.4 pp) to be considered almost practical.

5 Limitations and Threats to Validity

Our evaluation uses only one smartphone which limits how well our results generalize. We chose the Motorola phone as our test device because of the easy battery access so that others can easily replicate our setup and results. As it runs a mostly unmodified version of Android it reflects the intended behavior of the operating system. We verified this by running the DontKillMyApp [31] benchmark which schedules and later verifies various background operations. Our test device received a perfect score. Other vendors add custom battery saving techniques which can introduce bugs or restrict background services [31]. This applies also to iOS devices which are more restrictive on background activities. Therefore, our results may not directly generalize to other devices in terms of functionality and power consumption. Running background cover traffic over long periods of time might even be prevented by the operating system on some platforms without explicit intervention by users. Similarly, differences in modem hardware and configuration affect the obtained power measurements. However, we believe that results for devices with similar functionality are comparable, with newer devices generally being more energy efficient.

We chose GiffGaff as the mobile network provider as they offer pre-paid SIM cards. While it appears representative to us based on our observations, the results do not necessarily translate to other providers. For instance, the duration of tail-latency demotion is a parameter that is set by the network provider and can change over time. The GiffGaff configuration has a fairly short tail-latency which appears favorable for regular small messages. Therefore, the results for mix networks can be seen as a lower-bound estimate and we might expect it to be higher for other providers.

For our macro studies we use a public website hosted by a third-party. Changes to the website will inadvertently change the obtained absolute measurements. Likewise, mobile network conditions are not perfectly reproducible. For example, upgrades by the provider and testing in different locations will lead to slightly different numbers. As a mitigation, we always (re-)ran all VPN and Tor experiments together at the same location to ensure comparability of the results.

6 Conclusion

Our paper shows that there has been little attention to the viability of anonymity networks on smartphones. In particular, energy measurements are a blind spot in the evaluation of anonymity networks. This is concerning as smartphones have become our primary computing devices. We hope that our work motivates other researchers to evaluate their designs and implementations on real devices to determine their feasibility. Our open-source hardware setup and analysis tools reduce the barrier to do so and make results easier to compare.

Our evaluation highlights the dominance of radio transmissions for overall power consumption. At the same time we find that cryptographic operations have become negligibly cheap in the latest generation of devices. Hence, it is the scheduling of messages and their impact on latency that are critical when creating and evaluating new anonymity network designs. This is especially true for networks that provide unobservable communication through the use of cover traffic. We would find it interesting to see protocols that address this issue by reducing radio communication through batching of messages and using traffic scheduling that is more adaptable. However, doing so with minimal impact on the provided metadata privacy guarantees and perceived end-user performance is a challenging problem.

As cryptography only contributes little to the overall energy impact of mix networks, pre-computation (e.g. of the cover traffic messages) does not seem worth the additional complexity. However, this might be different for other kinds of anonymity networks or on embedded devices with tighter or more volatile energy constraints.

We believe that anonymity networks need to be practical on smartphones in order to reach widespread adoption. Tor works, but other designs that have become popular in academic discussion currently require too much energy to be practical in many cases. This opens up the opportunity for interesting new designs that take these constraints into account and explore solutions that work on mobile.

[1] We calculate $12\,\text{h} \times (0.8\,\text{pp/h} - 0.4\,\text{pp/h}) = 4.8\,\text{pp}$ using data from Table 4 in Appendix B. Similar for the other numbers.

Acknowledgements

We would like to thank the anonymous reviewers and the excellent shepherd. Daniel Hugenroth is supported by Nokia Bell Labs and the Cambridge Trust.

References

[1] Martin R Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. Collective information security in large-scale urban protests: the case of Hong Kong. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3363–3380, 2021.

[2] Sebastian Angel and Srinath TV Setty. Unobservable communication over fully untrusted infrastructure. In *OSDI*, volume 16, pages 551–569, 2016.

[3] Luca Ardito, Giuseppe Procaccianti, Marco Torchiano, and Giuseppe Migliore. Profiling power consumption on mobile devices. *ENERGY*, pages 101–106, 2013.

[4] Arm Limited. *Arm Instruction Set Reference Guide*, 2018. Version 1.0 (100076_0100_00_en).

[5] Ludovic Barman, Moshe Kol, David Lazar, Yossi Gilad, and Nickolai Zeldovich. Groove: Flexible metadata-private messaging. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 735–750, 2022.

[6] Aaron Carroll, Gernot Heiser, et al. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*, volume 14, pages 21–21. Boston, MA, 2010.

[7] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), 1981.

[8] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri De Ruiter, and Alan T Sherman. cmix: Mixing with minimal real-time asymmetric cryptographic operations. In *International conference on applied cryptography and network security*, pages 557–578. Springer, 2017.

[9] 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, 2015.

[10] Raymond Cheng, William Scott, Elisaweta Masserova, Irene Zhang, Vipul Goyal, Thomas Anderson, Arvind Krishnamurthy, and Bryan Parno. Talek: Private group messaging with hidden access patterns. In *Annual Computer Security Applications Conference*, pages 84–99, 2020.

[11] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE, 2015.

[12] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *2009 30th IEEE Symposium on Security and Privacy*, pages 269–282. IEEE, 2009.

[13] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *International Workshop on Information Hiding*, pages 293–308. Springer, 2004.

[14] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The Nym network (whitepaper), 2021.

[15] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[16] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, Dan Boneh, et al. Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. In *USENIX Security Symposium*, pages 1775–1792, 2021.

[17] Abram Hindle, Alex Wilson, Kent Rasmussen, E Jed Barlow, Joshua Charles Campbell, and Stephen Romansky. Greenminer: A hardware based mining software repositories software energy consumption framework. In *Proceedings of the 11th working conference on mining software repositories*, pages 12–21, 2014.

[18] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 225–238, 2012.

[19] Daniel Hugenroth. Android Support for Elliptic Curves (EC) in KeyPairGenerator, 2021. https://www.danielhugenroth.com/posts/2021_07_ec_curves_on_android.

[20] Apple Inc. Choosing Background Strategies for Your App, 2022. https://developer.apple.com/documentation/backgroundtasks/choosing_background_strategies_for_your_app.

[21] Google Inc. Android O prevents access to /proc/stat , 2017. https://issuetracker.google.com/issues/37140047#comment2.

[22] Google Inc. Optimize for Doze and App Standby, 2021. https://developer.android.com/training/monitoring-device-state/doze-standby.

[23] Google Inc. Power management restrictions, 2021. https://developer.android.com/topic/performance/power/power-details.

[24] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security*, pages 27–46. Springer, 2016.

[25] Stephan A Kollmann and Alastair R Beresford. The cost of push notifications for smartphones using Tor hidden services. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 76–85. IEEE, 2017.

[26] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Yodel: strong metadata security for voice calls. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 211–224, 2019.

[27] Andrea McIntosh, Safwat Hassan, and Abram Hindle. What can Android mobile app developers do about the energy consumption of machine learning? *Empirical Software Engineering*, 24(2):562–601, 2019.

[28] José A Montenegro, Mónica Pinto, and Lidia Fuentes. What do software developers need to know to build secure energy-efficient Android applications? *IEEE Access*, 6:1428–1450, 2017.

[29] Max Mössinger, Benedikt Petschkuhn, Johannes Bauer, Ralf C Staudemeyer, Marcin Wójcik, and Henrich C Pöhls. Towards quantifying the cost of a secure IoT: Overhead and energy consumption of ECC signatures on an ARM-based device. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6. IEEE, 2016.

[30] Steven J Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy*, pages 183–195. IEEE, 2005.

[31] Petr Nalevka and Jiří Richter. Don't kill my app!, 2022. https://dontkillmyapp.com/.

[32] NymTech. A Sphinx packet implementation in rust, 2021. https://github.com/nymtech/sphinx.

[33] NymTech. Nym Network Explorer a distributed power monitoring platform for mobile devices, 2022. https://explorer.nymtech.net/.

[34] Abhinav Pathak, Y Charlie Hu, and Ming Zhang. Where is the energy spent inside my app? fine grained energy accounting on smartphones with Eprof. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 29–42, 2012.

[35] Mike Perry and George Kadianakis. Tor padding specification, September 2021. https://github.com/torproject/torspec/blob/main/padding-spec.txt.

[36] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, August 2010. v0.34, http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf.

[37] Ania M Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The Loopix anonymity system. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1199–1216, 2017.

[38] Nachiketh R Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on mobile computing*, 5(2):128–143, 2005.

[39] Andrew Rice and Simon Hay. Measuring mobile phone energy consumption for 802.11 wireless networking. *Pervasive and Mobile Computing*, 6(6):593–606, 2010.

[40] Helena Rifa-Pous and Jordi Herrera-Joancomart. Computational and energy costs of cryptographic algorithms on handheld devices. *Future internet*, 3(1):31–48, 2011.

[41] David Schatz, Michael Rossberg, and Guenter Schaefer. Hydra: Practical metadata security for contact discovery, messaging, and dialing. In *ICISSP*, pages 191–203, 2021.

[42] Laurent Simon, Wenduan Xu, and Ross Anderson. Don't interrupt me while I type: Inferring text entered through gesture typing on Android keyboards. In *16th Privacy Enhancing Technologies Symposium (PETS)*, 2016.

[43] Texas Instruments. *INA219 Zero-Drift, Bidirectional Current/Power Monitor With I2C Interface (SBOS448G)*, 12 2015. Rev. G.

[44] The Guardian Project. Orbot: Proxy with Tor, 2022. https://guardianproject.info/apps/org.torproject.android/.

[45] The Tor Project. FAQ: Do i need both Tor Browser for Android and Orbot, or only one?, 2022. https://support.torproject.org/tormobile/tormobile-6/.

[46] The Verge. There are over 3 billion active Android devices, 2021. https://www.theverge.com/2021/5/18/22440813/android-devices-active-number-smartphones-google-2021.

[47] Robert Triggs. Fact check: Is smartphone battery capacity growing or staying the same? , 2018. https://www.androidauthority.com/smartphone-battery-capacity-887305.

[48] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[49] Matteo Varvello, Kleomenis Katevas, Mihai Plesa, Hamed Haddadi, and Benjamin Livshits. BatteryLab: a distributed power monitoring platform for mobile devices. In *HotNets '19*, 2019.

[50] Ekhiotz Jon Vergara, Simon Andersson, and Simin Nadjm-Tehrani. When mice consume like elephants: Instant messaging applications. In *Proceedings of the 5th international conference on Future energy systems*, pages 97–107, 2014.

[51] Ekhiotz Jon Vergara, Simin Nadjm-Tehrani, and Mihails Prihodko. EnergyBox: Disclosing the wireless transmission energy cost for mobile devices. *Sustainable Computing: Informatics and Systems*, 4(2):118–135, 2014.

[52] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 179–182, 2012.

[53] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114, 2010.

A Background Overhead

We show the execution of a test operation (100ms sleep) using WakeLock in Figure 12. Even while idle, the base power consumption is higher than when using the AlarmManager. Also, by holding the WakeLock, other background timers and checks can run. These are visible as the small peaks.

Figure 13 shows the execution of the same operation using AlarmManager. Outside the execution the baseline power drops to the technical minimum. However, for every execution we pay with an overhead to wake up (before *start*) and a period of time where the system delays going back to idle (after *start*). Also, note that the actual start of execution happens after the scheduled time *sched*.

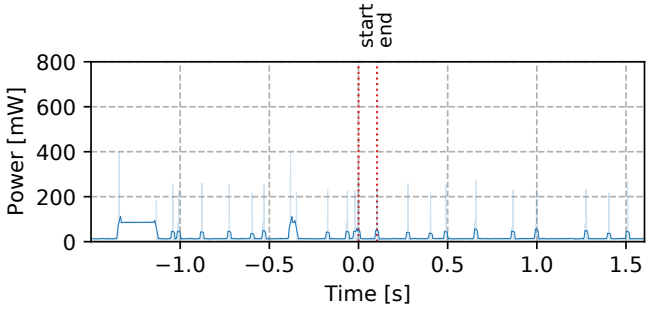


Figure 12: Execution of a test operation (100ms sleep) using WakeLock.

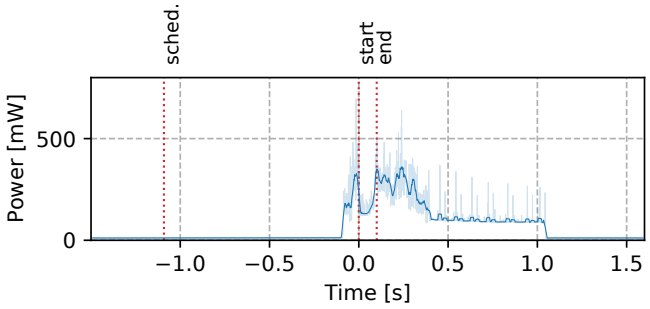


Figure 13: Execution of a test operation (100ms sleep) using AlarmManager.

B Measurements Data

This appendix contains the numerical results for the graphs in the main part of the paper. Tables 3 and 4 provide the detailed numbers from the macro studies on VPN and Tor, respectively, in Sections 4.4, 4.5, and 4.6. The battery life-time calculations use the a typical battery capacity of 8000 mWh (see §4).

C Literature Review

We summarize our literature review in Table 6 (last page). It contains the most significant papers related to our approach and experiments. We note that the most-recent study on cryptographic operations on mobile devices is from 2011.

D Additional Figures

We provide additional photographs of our setup in Figure 14.

Figure 14: Photographs showing the 3D printed battery holder and inlet.

	Network	Power[mW]	Battery[pp/h]
	Idle		
4G	Direct	29.3	0.4
4G	VPN_A	73.7	0.9
4G	VPN_B	135.3	1.7
WiFi	Direct	24.1	0.3
WiFi	VPN_A	32.0	0.4
WiFi	VPN_B	41.0	0.5
	Web (.com)		
4G	Direct	102.5	1.3
4G	VPN_A	202.3	2.5
4G	VPN_B	227.4	2.8
WiFi	Direct	48.6	0.6
WiFi	VPN_A	71.4	0.9
WiFi	VPN_B	81.0	1.0

Table 3: Average power consumption of *Direct* Internet access compared to using the two tested VPNs VPN_A and VPN_B for different scenarios. See Section 4.4

	Network	Power[mW]	Battery[pp/h]
	Idle		
4G	Direct	29.1	0.4
4G	Tor w/o Padding	61.1	0.8
4G	Tor w/ Red. Padding	71.2	0.9
4G	Tor w/ Full Padding	230.8	2.9
WiFi	Direct	25.0	0.3
WiFi	Tor w/o Padding	26.0	0.3
WiFi	Tor w/ Red. Padding	24.7	0.3
WiFi	Tor w/ Full Padding	84.5	1.1
	Web (.com)		
4G	Direct	122.1	1.5
4G	Tor w/o Padding	269.0	3.4
4G	Tor w/ Red. Padding	326.0	4.1
4G	Tor w/ Full Padding	378.1	4.7
WiFi	Direct	50.0	0.6
WiFi	Tor w/o Padding	111.9	1.4
WiFi	Tor w/ Red. Padding	106.9	1.3
WiFi	Tor w/ Full Padding	121.2	1.5
	Web (.onion)		
4G	Tor w/o Padding	369.1	4.6
4G	Tor w/ Red. Padding	360.2	4.5
4G	Tor w/ Full Padding	390.3	4.9
WiFi	Tor w/o Padding	90.7	1.1
WiFi	Tor w/ Red. Padding	76.6	1.0
WiFi	Tor w/ Full Padding	172.3	2.2

Table 4: Average power consumption of *Direct* Internet access compared to using *Tor* for different scenarios. See Section 4.5.

Operation	PDA 2011 [40]	ARM IoT 2016 [29]	Ours
Gen RSA-1024	1186.79	×	116.47
Sign RSA-1024	24.05	×	1.88
Verify RSA-1024	1.35	×	0.34
Gen RSA-2048	×	×	116.47
Sign RSA-2048	102.08	×	6.22
Verify RSA-2048	3.17	×	0.50
Gen EC-224	29.05	24	1.11
Sign EC-224	38.24	27	1.43
Verify EC-224	48.71	29	1.52

Table 5: Comparison of energy costs for cryptographic operations between two previous studies and our results. All data in mJ. Where multiple data points are available we chose the smallest one. × incidcates missing data: the 2011 paper does not include the key generation for 2048-bit RSA and the 2016 paper only measured EC operations. See Section 4.1.

Year	Title	Authors	Summary
	Papers with direct measurements		
2005	*Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks.	Wander et al.	Measured radio communications and cryptographic operations on an ATMega128.
2006	A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols	Potlapally et al.	Measured cryptographic operations and SSL on a PDA.
2010	Measuring mobile phone energy consumption for 802.11 wireless networking	Rice et al.	In-depth WiFi power analysis on an Android smartphone.
2010	*An analysis of power consumption In a smartphone	Carrol et al.	Measures various energy components on multiple Android devices.
2010	Exhausting battery statistics: understanding the energy demands on mobile handsets	Vallina-Rodriguez et al.	Measured battery of devices of volunteers.
2011	Computational and Energy Costs of Cryptographic Algorithmson Handheld Devices	Rifa-Pous et al.	Measured cryptographic operations on a PDA.
2012	*A close examination of performance and power characteristics of 4G LTE networks	Huang et al.	In-depth 4G power analysis on mobile devices.
2013	Profiling Power Consumption on Mobile Devices	Adrito	Measured different scenarios on Android phones (calls, MP3 playback, . . .).
2014	GreenMiner: A hardware based mining software repositories software energy consumption framework.	Hindle et al.	Measured apps in a highly-automated setup.
2016	Towards quantifying the cost of a secure IoT: Overhead and energy consumption of ECC signatures on an ARM-based device	Mössinger et al.	Measured ECC signatures on IoT devices.
2019	What can Android mobile app developers do about the energy consumption of machine learning?	McIntosh et al.	Measured different ML algorithms on Android.
2019	BatteryLab: A Distributed Power Monitoring Platform For Mobile Devices	Varvello et al.	Framework for remote controlled measurements and measured browser efficiency
	Papers which create models		
2010	*Accurate online power estimation and automatic battery behavior based power model generation for smartphones	Zhang et al.	Created PowerTutor which can estimate the power consumption of apps.
2014	EnergyBox: Disclosing the wireless transmission energy cost for mobile devices.	Vergara et al.	Created EnergyBox which can estimate the power consumption for 3G and WiFi based on captured packets.
	Papers which use models		
2012	*Where is the energy spent inside my app? fine grained energy accounting on smartphones with eprof.	Pathak et al.	Enhanced existing models by tracking which app components are responsible.
2013	How much energy can we save from prefetching ads?: Energy drain analysis of top 100 apps	Chen et al.	Used Pathak et al.'s work to show that pre-fetching of Ads has negligible impact.
2014	When Mice Consume Like Elephants: Instant Messaging Applications	Vegara et al.	Used EnergyBox to compute savings of improved schedulers for Instant Messaging apps.
2017	The Cost of Push Notifications for Smartphones using Tor Hidden Services	Kollmann et al.	Used EnergyBox to investigate the feasibility of push notifications over Tor.
2017	What Do Software Developers Need to Know to Build Secure Energy-Efficient Android Applications?	Montenegro	Used PowerTutor to compare different cryptography libraries on Android.

Table 6: Literature grouped by methodology and sorted by year. Papers marked with * have more than 500 citations.