Maintaining Integrity and Reputation in Content Offloading

Torsten Zimmermann, Jan Rüth, Hanno Wirtz, Klaus Wehrle
Chair of Communication and Distributed Systems, RWTH Aachen University
{zimmermann, rueth, wirtz, wehrle}@comsys.rwth-aachen.de

***Abstract*—The growing demand for mobile content has increased the burden on cellular network providers. To this end, mobile content offloading approaches aim to offer a relief of overloaded cellular network infrastructures via local content exchanges between mobile devices. A core assumption of proposed approaches is the voluntary, honest, and altruistic participation of devices or their owners. This dependency offers malicious participants an avenue of mounting denial-of-service attacks by lying about the content they have or by providing forged content, thereby negating the principal advantages of mobile offloading. Furthermore, without mutual authentication between the participants, there are no means of identification and thereby no chance to report or to stop abuse.**
In this paper, we thus propose MIRCO, an approach that adds integrity protection to the offloaded content, authentication for participants, and a commitment to the exchange of content between devices. We embed the aforementioned techniques into the offloading process and implement and evaluate our approach using Android smartphones with regard to time and energy overhead. Our results show that MIRCO only adds negligible overhead to content exchanges, thus enabling a feasible and accountable content offloading approach.

I. Introduction

Mobile broadband usage has experienced a massive growth [1] which increases the burden for the cellular operators, forcing them to adapt their networks to still provide adequate service for the users. Solutions include adding more resources in the form of cellular base stations [2] or adding additional caches at the edge of the network [3], in order to reduce the load in the provider's backbone. While the first solution implicates a significant amount of costs and maintenance effort, the latter still suffers from the need to send the requested content to each individual user [3], [4]. Therefore, alternative solutions such as mobile content offloading have been introduced. Mobile content offloading approaches offer the possibility to pre-cache popular content [4], [5] on mobile devices and additionally leverage opportunistic Device-to-Device (D2D) communication for further data distribution [6] on a local scale.

While mobile offloading thereby offers a relief of overloaded cellular network infrastructures via content exchanges between mobile devices [4], [5], [7]–[11], a core assumption is the voluntary, honest, and altruistic participation of devices or their owners, respectively. This is because, once content has been offloaded from the Internet, subsequent D2D communication offers no control, as originally offered by the original content provider, over the authenticity and integrity of content. Hence, devices that partake in content offloading need to rely on providing devices to honestly announce their available content and to actually provide the eventually requested content. This dependency then offers malicious devices an avenue of mounting denial-of-service attacks by lying about the content they have or by providing fake or harmful content, thereby negating the principal advantages of mobile offloading as well as wasting communication and energy resources in the process.

Within this scope, multiple solutions facilitate actual offloading of content from the Internet [4], [5], [9], expression of content interests [12], and local connectivity and content exchanges between devices [7], [8], [13]. In contrast, ensuring content authenticity and integrity, as well as correct behavior by participant remains an open problem. This is because applying traditional means of content protection in mobile offloading, i.e., end-to-end content encryption and certificate-based authentication of the provider at request time, is infeasible as no end-to-end connection between provider and requester exists. Similarly, group encryption of content to all participants in an offloading scheme does not hinder malicious participants from altering and/or lying about content. Moreover, without authentication between participants, identification of misbehaving participants and ultimately the ability to exclude them is impossible. We argue that these functionalities are central to the adoption and acceptance of offloading approaches in real-world applications.

In this paper, we present MIRCO, an approach to maintain integrity and reputation in content offloading. We therefore enhance the general design of current content offloading approaches by 1) a trusted signing proxy that attaches a cryptographic signature to content that is offloaded from the Internet to the providing mobile device, 2) authentication of participating devices and cryptographic commitment to the content they agree to exchange, 3) verification of received content authenticity by way of the aforementioned trusted signature, and 4) a reputation system that chronicles the correctness of device behavior and content exchanges. We design MIRCO such that it seamlessly embeds itself in the design of current architectures [9]–[11], [13] and fosters the fundamentally collaborative nature of mobile offloading in letting participants report malicious behavior by both requesting and providing devices.

The rest of this paper is structured as follows. In Section II, we motivate the requirements for MIRCO by presenting current approaches and building blocks for content offloading. In Section III, we present our design of maintaining content integrity and reputation through user authentication. Moreover, we demonstrate how this integrates into common offloading processes. We implement MIRCO on Android smartphones and evaluate its performance with regard to time and energy consumption in Section IV and conclude in Section V.

II. Related Work

MIRCO enhances common approaches and building blocks that enable mobile content offloading, like the initial content distribution or the local search for content and its exchange. We motivate the requirements for our design based on the characteristics of existing approaches and possible limitations.

A. Initial Caching of Content

For network operators, an important task in mobile offloading is the identification of content that relieves their networks the most. Attributes of such content are *i)* the popularity, based on the requests, and the *ii)* tolerance to delay, i.e., non-interactive. Either, content is classified as popular on the device locally based on individual user request [5] or such a classification is performed at the cellular provider on a larger scale [3], [4], [14]. In addition, an actual content provider could also share statistics about requested and delivered content. Nevertheless, as long as content is self-contained and non-interactive it may be cached by participating devices.

Approaches that build upon global content classification can pack popular content [4] into bundles and send these to the mobile stations on a regular basis, e.g., via a cell multicast or store them on caches located at the edges of the network. Using a local, i.e., user specific, classification of popular content, approaches like *IMP* [5] try to optimize the download of this content, by taking battery lifetime and connection type, e.g., Wi-Fi or cellular, into account. In this individual case, such content may also include emails. Systems that also target the further offloading of popular content are presented in [9]–[11], [15]. To this end, the authors propose to offload popular non-interactive content to either an initial set of mobile client devices or to all devices currently available in a cell, thus serving as helpers for further distribution. In [15], the authors propose a restriction to a trusted set of distributers, and forbid the further distribution by others.

Using a trusted set of initial distributers might mitigate the possibility to spread forged content, but may limit the overall availability of content. When each participant is allowed to distribute content, there is the need to check the authenticity and integrity of content. To allow for such checks, a trusted entity has to add additional information to content, e.g., such as authentication signatures as proposed in [16]. Moreover, clients need some form of authentication or reputation, e.g., as used in DTN approaches [15], among themselves to check, if they are communicating with a legit participant and to report possible misbehavior.

B. Locating and Obtaining Content

Locating and receiving content without the assistance of a third party has been covered in approaches targeting mobile opportunistic and Delay Tolerant Networks (DTNs). Approaches presented in this area either focus on optimizing content querying in previously established ad-hoc networks [12], or on combining the search for content on a local scale with the establishment of content centric networks on demand [13].

Approaches like *DataSpotting* [8] and *ICON* [7] add a specific entity located at the cellular provider that is responsible for locating and matching content. Clients report their location, their content interests, and the content they provide to this entity. If there is a match between two clients and if they are within D2D communication range, this entity triggers the respective clients via the cellular network to establish a local connection, e.g., via Wi-Fi, allowing them to exchange the content. As all control traffic is sent via the cellular connection, the cellular provider can identify clients.

However, neither the local approaches nor the approaches assisted by additional entities are either capable of checking if a client still is or was in possession of the requested content, or keeping track if the initiated exchange is successful.

In [17], requesting clients simultaneously contact a *Central Dissemination Manager (CDM)* via their cellular connection and try to obtain content through opportunistic contacts. If the local lookup is not successful within a certain timeout, the content is delivered via the CDM. In case of a successful local lookup, the requesting client acknowledges the exchange to the CDM. Moreover, the requesting node saves the content for a so called *sharing timeout*. The combination of this timeout and the acknowledgment of content exchanges allows the CDM to keep track of the content availability in the network. Again, there is no mechanism that checks the authenticity and integrity of the exchanged content automatically. Without such a check, the records kept at the CDM might get distorted.

We address the shortcomings in terms of integrity protection and reputation within the presented related work and show how we overcome these in MIRCO in the following section.

III. MIRCO Design

The overall goal of MIRCO is to maintain integrity and reputation in content offloading. To this end, we integrate the necessary support in the typical steps of mobile content offloading (see Section II), namely the initial distribution of content (cf. Figure 1) and subsequently the content exchange between mobile devices (cf. Figure 2).
For the initial content distribution (Figure 1), client (**A**) requests the desired content ($\mathbf{ID_X}$) via her cellular connection (1). Instead of directly requesting content from the content provider (**CP**), the content is requested over a secure connection via an additional trusted entity, the signing proxy (**P**). On behalf of the requesting client, this signing proxy then obtains the content ($\mathbf{Item_X}$) at the content provider (2-3). P extends the content with a content description ($\mathbf{CD_X}$) that will be used to match content to requests on a local scale. After the resulting bundle is signed ($[\ldots]_\mathbf{P}$), P sends it to the client (4). By extending the bundle with a signature, clients that obtain this content via local Device-to-Device (D2D) communication can directly check the content validity.

The local distribution of content to other clients (Figure 2) represents the second stage within common offloading approaches. A requesting client (**B**) sends the actual content request, a locally generated unique transaction ID ($\mathbf{T_{ID}}$), and her authentication credentials ($\mathbf{Cert_B}$) in a signed message ($[\ldots]_\mathbf{B}$) via a previously established D2D connection (1). To confirm this request, the providing client (**A**) sends a signed reply ($[\ldots]_\mathbf{A}$) containing her authentication credentials ($\mathbf{Cert_A}$) together with the previously received T_{ID}, the matching Content Description ($\mathbf{CD_X}$) as well as the signature of the request (2). We call this reply a *commitment* to this transaction. The

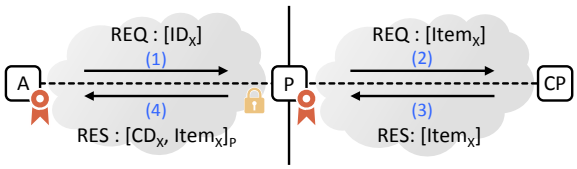


Fig. 1. After mutual authentication using certificates, the registered user A requests content at the signing proxy P (1). On behalf of A, P obtains the content from the content provider CP (2-3). After P has added the content description CD_X and has signed ($[\ldots]_P$) this bundle, it is sent to A (4).

providing client A signs the content bundle along with the T_{ID} and sends them to B (3). Because this content bundle is signed by P, the integrity of the content can be checked automatically. As there could be a multitude of different signing proxies, the certificate (**Cert$_P$**) of the respective signing proxy is included in this message. Finally, both clients issue a receipt for the successful transaction and thereby cryptographically commit to this exchange (4-5).

In the following, we focus on the necessary prerequisites that enable MIRCO. Moreover, we provide a detailed description of the aforementioned steps of the offloading process. Finally, we show how MIRCO enables reputation by reporting content exchanges between clients and how we can monitor and detect the misbehavior of malicious clients.

A. Assumptions & Prerequisites

The entities involved in the mobile content offloading scenario we envision in this paper are the *content provider*, an additional trusted entity in form of a *signing proxy*, and the *mobile clients*, as depicted in Figure 1. While we do not demand any changes to the content provider, we require additional, previously established, credentials for the authentication of the other entities as well as content signatures. The latter could either be computed on demand or proactively by the signing proxy. To allow clients and infrastructure elements, e.g., the signing proxy, to mutually authenticate in MIRCO, certificates are deployed. Therefore, we use a Public Key Infrastructure (PKI) containing two Certificate Authorities (CA), where each CA takes the role of issuing certificates to the aforementioned entities, respectively. For the certificates and keying material, we follow the NIST recommendations [18] and use Elliptic Curve Cryptography (ECC)[1] with a key length of 256 bit and SHA-256 as basis for the digital signatures. We choose two CAs to allow clients to easily distinguish between the two groups of entities. To enable a reputation system, clients need to register to be identified within the system. Thus, we equip clients with a certificate as well as the respective certificate chains during a registration process.

B. Obtaining Content via the Signing Proxy

Before requesting the actual content, a requesting client A and the signing proxy P perform mutual authentication using the previously established credentials by establishing a TLS connection (cf. Figure 1). Client A then requests (1) the content, we envision using an identifier (ID_X) that directly addresses content or that can be matched against a set of tags used to describe content.

[1] used curve is NIST P-256

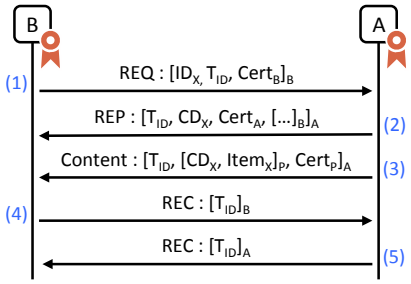


Fig. 2. Using local D2D communication, B sends the request containing the content ID (ID_X), a unique transaction ID (T_{ID}) and her certificate as signed ($[\ldots]_B$) message to A (1). The reply contains A's certificate together with the chosen T_{ID}, the content description (CD_X) as well as the signature of the request and is signed ($[\ldots]_A$) by A (2). Now, A sends the requested content bundle to B (3). Finally, both devices acknowledge the transaction by issuing receipts, which are signed by the respective clients (4-5).

When P can handle the client's request, it requests (2) the actual content according to ID_X on behalf of A at the content provider CP. After P has finished the content download (3), it creates a content bundle that also encloses the content description CD_X that matches ID_X. Fields contained in the description could be the *origin* or *type*, declaring if this is a web page or multimedia file, or a *category*, that may include multiple tags describing the topic, e.g., `news`, `sports`, or `entertainment`. This allows clients in the system to indicate and search for requested content semantically. As proposed in [19], the client could query for `web://news/cnn/business` or `media://youtube/top5`, or even more abstract, `web://news/sports`. The bundle is signed by P as a whole and then sent to client A (4). The content description offers two major benefits for our system. On the one hand, a client that requests content via the local D2D network might thus query more extensively, i.e., a client may express interest in content in a less specific fashion. On the other hand, it enables the client to check offloaded replies to her request in an automated fashion by matching content description CD_X within the response. Although this initial signing process adds a certain overhead compared to a regular download, it allows clients to directly verify the validity of content. This limits the distribution of forged content and the possibly resulting waste of time for honest clients, which we discuss in the following.

C. Client-to-Client Content Exchange

As client A now is in possession of content and a respective signature, it becomes a providing client and other clients now have the chance to request this bundle using local D2D communication. We assume the presence of a mechanism to allow for the spontaneous creation of a local Wi-Fi network. Among the existing techniques, clients could integrate the content request in a network request [13] or periodically switch between a *listen* (provider) and *discover* (requester) state [20].

In the following, we describe the necessary steps for a complete D2D content transaction. The requesting client B addresses the desired content by an identifier as described in Section III-B, depicted as ID_X in Figure 2. Moreover, B has to choose a unique transaction ID (T_{ID}) that is utilized to identify the whole transaction afterward. To this end, we use a 16 Byte (pseudo)-random Universally Unique Identifier (UUID) [21]. Furthermore, B adds her own certificate $Cert_B$ and finally signs the whole request (1). Upon reception, A first

checks the validity of $Cert_B$ with the certificate chain. If B is indeed showing a correct certificate, A checks the integrity of the request. On success, A performs a local content lookup, otherwise, A terminates the local connection. If the requested ID_X matches to a description CD_X of a content, A replies with the same T_{ID}, the matched content descriptor CD_X, the signature of the request, and her certificate $Cert_A$ (2). We incorporate the request's signature to mitigate the effect cooperative malicious clients may have on the system, which we discuss in detail in Section III-E. Please note that this message is also signed. As the request and the reply contain the transaction ID T_{ID} and are signed by the respective parties, we call this a *commitment* to this transaction. If the received content description indeed matches the requested content identifier, B now awaits the content, as the providing client A has at this point committed herself to this transaction.

Subsequently, A sends the content bundle, i.e., the requested content and its description signed by P, as well as the T_{ID} to B (3). When B has completely received this message, she immediately checks the integrity and if the content was truly issued by P. Moreover, she checks A's signature to verify the commitment to this transaction as well as if the description matches the one contained in the reply. If these checks turn out satisfactory, B issues a receipt to A in order to acknowledge the success of this transaction (4). Afterwards, A replies with her own receipt (5). Both receipts contain T_{ID} and are signed by the respective client. As a final step, both clients store the request, the reply, and both receipts for reporting which we discuss in the following.

D. Reporting Transactions

Up to now, we have shown how content is acquired and distributed in MIRCO. While this allows partaking clients to directly check the content's validity, we still need a mechanism to identify both altruistic and misbehaving clients. Therefore, we introduce a reporting system for transactions. To report a particular transaction, the clients have to send the respective *request*, the *reply*, as well as both *receipts* to the reporting system. It is not necessary, that this happens immediately after the transaction. Clients may wait until they again have a better connection available, e.g., LTE or Wi-Fi. The reporting entity itself could be co-located with the server or be realized as a logical single service for scalability reasons.

In case of a full transaction, i.e., all messages are present, the reporting system simply checks the validity of these with the help of the respective public keys. Notably, it is sufficient if only one of the clients sends the messages to the provider as these contain all necessary information and are signed by the respective clients. If these checks turn out positive, i.e., the request, reply, and receipts can be matched successfully, the reporting system stores these to record successful transactions.

However, if the content has been received by client B (cf. Figure 2) and B has already checked the validity, it might happen that the receipt issued by B does not reach A. This may be caused by connection problems or a client moving out of the D2D communication range. Following the protocol, A will not send her receipt to B. However, the transaction can still be reported as a success. Eventually, B notifies the reporting system about this transaction, by sending her request, the reply, and only her receipt. Because B acknowledges this transaction by showing her receipt to P, we count this transaction as valid, in case that all necessary checks are correct. In contrast, it is not sufficient if A only sends her copy of the request and her reply. In that case, P either has to wait for B to send her receipt or to explicitly request the receipt from B. Thus, a providing device, in this case A, cannot report a transaction as successful without the confirmation of the requesting device.

TABLE I. EXEMPLARY INFORMATION STORED IN THE REPORTING SYSTEM FOR VARIOUS CLIENTS.

User	∅ Requests/d	∅ Transactions/d	Success	Rating
Alice	25	80	0.87	AAA
Bob	17	42	0.56	A
Mallory	12	79	0.09	D

We envision the reporting system to be accessible by registered users and make the status of transactions transparent. The data available at the reporting system could contain information as depicted in Table I. An envisioned reporting system could store the average number of requests to the signing proxy as well as the average number of transactions between devices on a per day basis. Moreover, the success rate of the latter can be calculated based on the reporting, resulting in a per user rating. In the next section, we will focus on how MIRCO monitors and detects potentially misbehaving clients and which actions are taken.

E. Monitoring and Detecting Abuse

The altruistic and correct behavior of clients is a crucial ingredient for mobile content offloading. In the following, we sketch different situations where clients are misbehaving and thereby try to harm mobile content offloading and discuss how we handle these situations in MIRCO.

Assume the following situation: after the request/response exchange (cf. Figure 2) client A sends content to B that is correctly signed by P and A but does not match the requested content. After the download is finished, B compares the content description and will recognize the mismatch. In this case, B reports this transaction as incorrect by sending the request, the reply as well as T_{ID}, CD_x, both signatures from the content header, and the hash of the content itself to the reporting system. Since A committed herself to this transaction, this will be marked as a failed transaction caused by A. If A adapts the content description but sends mismatching content to B, the check of the signature issued by P will fail. As in the previous situation, if A has signed the messages correctly, B will send the respective messages and header fields to the reporting system.

Furthermore, misbehaving clients could simply not send specific messages and thereby harm the system. For example, after the request and the reply have been sent, client A refuses to send the content or sends random bytes. In that case, client B has the chance to inform the reporting system about this incident. However, B can only prove that she has communicated with A, i.e., by showing the request and the reply, but cannot prove that A has not sent the content. The reporting system will record this, but not directly take actions against A, as this situation could also been caused by connectivity problems. Therefore, in MIRCO we recommend to leverage majority voting using the reporting system. If such an incident is reported frequently, the

system will ultimately inform A about the incident and could mark A as harmful.

To limit the possibility for a set of clients to work together to incorrectly force the reporting system to exclude a particular client, they have to at least prove that they communicated with this client. Therefore, we include the signature of the request in the reply (cf. Figure 2) to prevent clients to replay a copy of a request-reply pair to the reporting system, which would be possible if we only rely on the transaction ID.

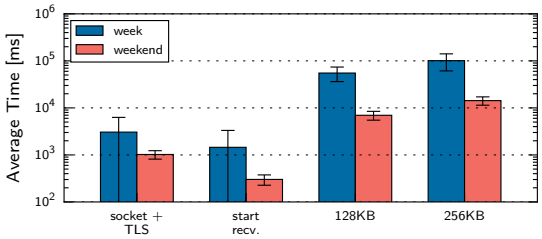
If a client is judged as harmful to the system, i.e., the rating falls below a certain threshold, there are different possibilities to sentence her, e.g., a time-penalty or a complete banishment from the system. To achieve both types, we suggest either to limit the *validity lifetime* of the issued client certificates, e.g., by simply setting the *Not Before* and the *Not After* fields accordingly, or to use Online Certificate Status Protocol (OCSP) stapling [22]. Hence, clients in MIRCO need to *refresh* the validity of their certificate on a regular basis. To achieve this, the client CA renews the respective fields of the client certificate and resigns it. Thus, communicating clients can directly check if the opposing certificate is still valid. Before the lifetime expires, the client needs to contact the respective CA to refresh her certificate. If the reporting system has marked this client as misbehaving, the CA can either refuse to refresh the certificate's validity for a certain amount of time or completely exclude her from the system.

Additionally, the individual user rating, as presented in Table I, could also be incorporated in this refreshment process. Integrating the actual user rating into the certificate would allow other clients to choose whether they want to communicate with a client that is not above a certain rating or to limit the content size they would be willing to download from such a client.

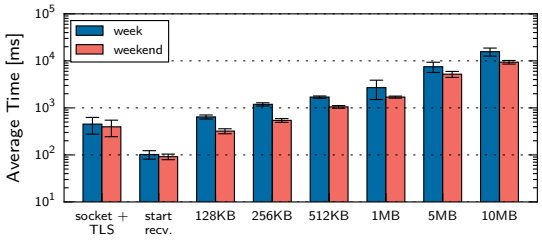
IV. Evaluation

To evaluate the feasibility and applicability of MIRCO, we implement the aforementioned clients and the signing proxy. Our evaluation covers the time for the initial content distribution between the client and the signing proxy (Section III-B), as well as the time needed for the content exchange between two devices (Section III-C). Furthermore, we show that our design only adds a minimal overhead on top of typical content offloading approaches, thus verifying our design decisions while protecting participants from misuse and possible attacks. To classify the impact on mobile clients with regard to energy usage, we measure the power consumption for various communication modes, i.e., 2G, 3G, LTE, and Wi-Fi, when using mobile offloading.

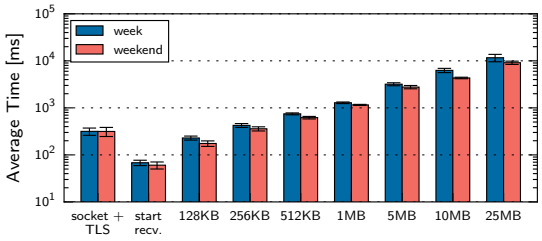
We implement the clients on two Nexus 5 devices running Android 5.1.1. As cryptography library, we use *Spongy Castle*[2], that extends the Android Bouncy Castle library with the support for Elliptic Curve Cryptography. The Nexus 5 is equipped with a 2.26 GHz CPU, a Broadcom BCM4339 5G Wi-Fi Chip and a Qualcomm WTR1605L cellular transceiver. One of the devices is endowed with an LTE-capable SIM card of a major European cellular provider. The signing proxy functionality is realized on a Ubuntu Desktop 14.04 machine with an Intel i7 2.93 GHz CPU and 4GB RAM. We base our server implementation on



(a) Time measurements for a 2G connection.



(b) Time measurements for a 3G connection.



(c) Time measurements for a LTE connection.

Fig. 3. Time measurements for the respective steps of requesting and receiving content at the signing proxy in different connection modes during working hours and the weekend. Values shown are the average and the standard deviation. Please note the logarithmic scale.

Twisted[3] and add the additional functionality, i.e., elliptic curve cryptography and signature handling, via the *M2Crypto*[4] toolkit for Python.

A. Timing

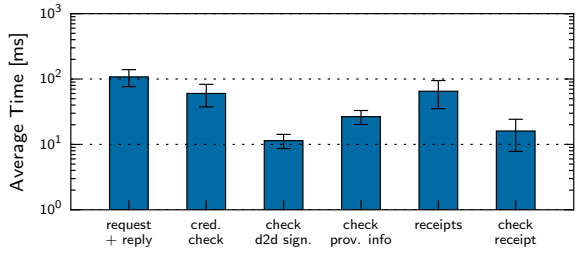
In the first evaluation scenario, we connect to the cellular network in different modes, i.e., 2G, 3G, and LTE. Subsequently, the device requests content of varying sizes (128 KB to 25 MB), representing content such as single pictures, whole websites, or videos, at the signing proxy. This request process follows the steps depicted in Figure 1. In our evaluation setup, the requested content is located directly at the signing proxy and we thereby omit the initial download at the actual content provider thus reflecting a well established cache. This process is measured in the parking lot of our institute building located just outside the city center in a suburban like area. We conduct 30 complete runs for each connection mode and for a set of content sizes respectively. As cellular network performance may vary with the number of participants, we repeat the experiments during working hours and on the weekend to assess the implications for mobile content offloading.

The goal of this evaluation is twofold, first, we want to show that the overhead introduced by the envisioned content distribution in MIRCO is negligible in comparison to the download time, second, this evaluation serves as a baseline for

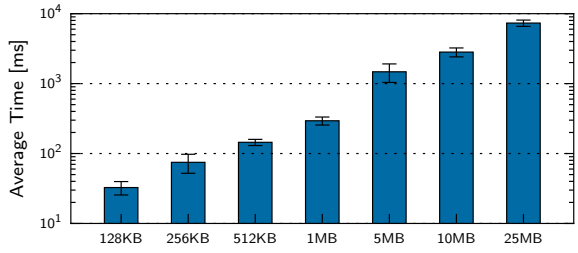
[2]https://rtyley.github.io/spongycastle/

[3]https://twistedmatrix.com

[4]https://pypi.python.org/pypi/M2Crypto



(a) Time results for MIRCO specific message exchanges and computations, resulting in a total of 287 ms.



(b) Time for the actual content download using Wi-Fi.

Fig. 4. Time measurements for the respective steps of a content exchange using a D2D connection realized via Wi-Fi. Values shown are the average and the standard deviation. Please note the logarithmic scale.

regular cellular downloads to which we eventually compare the D2D offloading approach. To this end, we measure the time it takes to create a connected socket, i.e., including the mutual authentication between the client and the signing proxy, the time from the content request to the start of reception, as well as the time for the whole content download. The request to the signing proxy contains a content identifier, which we set to 50 B for this evaluation. Together with the content, the signing proxy prefixes a content header containing the cryptographic signature of 80 B as well as a content description set to 80 B. Please note that additional messages fields in the request and the server's response, e.g., type, length, typically consume less than 20 B. Figures 3(a) - 3(c) show the average and standard deviation of the time requirement of each step.

As expected, the presence of more people and thereby devices, during the week has a greater influence on a 2G (EDGE) connection than on a 3G or LTE connection. The time for the content downloads of 128 KB and 256 KB in the 2G network increases from 7 s on the weekend to 55 s during working hours and from 14 s to 100 s respectively. In addition, the time for the connection setup increases from 1 s on the weekend to 3 s during the week and the time until the download starts (start recv.), i.e., sending a content request and waiting for the first bytes of the reply to arrive, takes 300 ms on the weekend and increases to 1.4 s when the network becomes populated during the week.

The download times for the 3G connection range from 640 ms to 16 s (128 KB to 10 MB) during the week and from 320 ms to 9 s during the weekend respectively. In both cases, i.e., the week and the weekend, the connection setup time is around 400 ms and time until the start of the download around 100 ms. In comparison, the LTE connection shortens the download time down to 230 ms for 128 KB and down to 12 s for an even bigger 25 MB content during the week. On the weekend the same content takes only 175 ms and 9 s to download respectively, thereby also showing that LTE suffers the least from crowded networks. The setup times and the time until the download starts are around 310 ms and 60 ms respectively, thus showing an even smaller roundtrip time. In summary, we can observe that for all modes, the initial connection setup is either always or quickly dominated by the download time.

We put such emphasis on these times, as they again motivate the use of mobile content offloading itself. 3G and LTE already offer a high performance, however if their service is disrupted, either due to an overload or to no coverage, the alternative 2G offers nearly no performance at all. Thus, a relief through a local high throughput link will improve the overall system performance as we will show in the following.

The second evaluation scenario comprises time measurements of the content exchange between two devices using MIRCO, following the scheme presented in Figure 2. Most strikingly, this also demonstrates the implications of abuse in mobile content offloading. Without integrity protection and mutual authentication as offered by MIRCO, malicious entities may introduce significant time overhead to honest participants by lying about the presence of content. For the evaluation of this scenario, we place the two Android devices at a distance of 15 m outside of our institute and let the device that possesses the content open an 802.11n Wi-Fi hotspot, announcing a rate of 73 Mbps. Subsequently, we let another device request content of varying size. As in the previous evaluation, we conduct 30 runs for each of these sizes. Figures 4(a) - 4(b) show the result for the additional steps of the D2D content exchange introduced by MIRCO and the time for the actual content download respectively.

In MIRCO, a client connects to a communication partner, requests content and obtains the first part of the reply in around 108 ms (cf. Figure 4(a) request + reply). To verify the content owner's response, again including certificate and signature, the requester needs roughly 60 ms (cred. check) to perform the same operations as the content distributor before. When the requesting device has received the content, it checks the received transmission signature with the previously validated certificate in about 11 ms (check d2d sign). Afterwards, it validates the included certificate of the signing proxy against the respective certificate chain and ultimately checks the content's signature using this certificate, which in total takes around 27 ms (check prov. info). Moreover, both devices issue a receipt to each other which takes another 65 ms (receipts) on average. Eventually, the requesting device checks the receipt from the communication partner within 16 ms (check receipt).

In our evaluation, the request message has a payload of 756 B. Besides control fields, it contains the 16 B transaction ID, an 80 B signature, the identifier we set to 50 B and the requesting clients certificate of size 600 B. The reply to this request has a size of 867 B, containing similar information as the request (cf. Figure 2). Instead of the identifier, we incorporate a fixed content description of size 80 B for this evaluation. In addition, the reply contains the signature of the request. The content header sums up to 881 B, again containing fields like the content description and the transaction ID. Furthermore, it incorporates the signing proxy's certificate and the respective signature. Finally, both receipts each have a size of 102 B. Given the fixed values for the content request and the content description, that might vary in reality, this sums up to a total of only 2708 B additional payload that is sent beside the actual content.

The total time overhead of the additional messages is comparable to the time of establishing a TLS channel and getting the first bytes. In the previous evaluation it took 374 ms to establish a channel over the LTE network on the weekend, however, with the additional authenticity and mutual receipt exchange that MIRCO employs, we are on par with a total of 287 ms. Ultimately, the size of the content will dominate the transmission time, Figure 4(b) shows the times for the actual content download. The times for these transfers range from 33 ms to 7.3 s (128 KB to 25 MB). Thus, in all tested setups, a local Wi-Fi channel outperforms the cellular networks even though LTE closes the gap, thereby showing the feasibility of mobile content offloading itself. However, without the features offered by MIRCO, a local D2D exchange is still vulnerable to malicious entities providing forged content, thereby rendering these advantages useless.

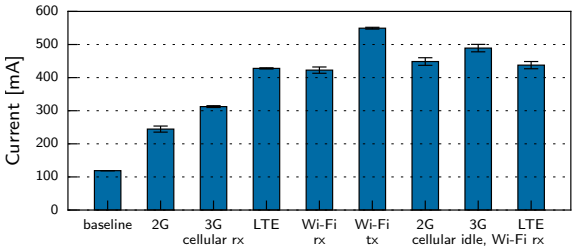
Still, a D2D channel needs to be bootstrapped beforehand, referring to the results shown in [13], discovery of communication partners and the establishment of a D2D network with respect to 802.11 security requirements takes less than 4.5 s on commodity smartphones. Within this time, an LTE connection in our test setup already downloaded around 4 MB. Nevertheless, when reconsidering that mobile content offloading tries to overcome the lack of a fast cellular connection or no connection at all, we still benefit even with such a delay. For example, a slow 2G connection only connected its socket and starts receiving the first bytes within this time, allowing MIRCO to overtake 2G within fractions of a second.

These results show that adding the ability to monitor and detect abuse of D2D connections only introduces a comparatively small amount of traffic as well as time overhead. Thus, our implementation of MIRCO demonstrates the feasibility and real-life applicability of incorporating integrity protection for content and authentication of participants.

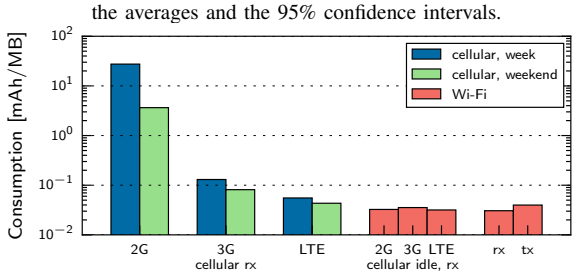
B. Power Consumption

Complementary to our evaluation of time, we also conduct measurements for the power consumption of the smartphone in various connection setups. On the one hand, we want to put the energy demands of a local D2D connection into perspective to cellular data exchange. On the other hand, we want to investigate the burden, i.e., energy consumption, honest participants are willing to take and that malicious participants may force on others. Therefore, we replace the battery of one smartphone with *PowerGraph*[5], a measurement tool developed at our institute to measure the current drain of a connected device. Internally, PowerGraph senses the current over a measurement shunt that is serially connected to the smartphone, while simultaneously acting as a power supply to deliver 3.8 V operation voltage. We set the smartphone to different modes, i.e., type of cellular connection, download a sufficiently dimensioned file and sample the signal with a rate of over 7 kHz for a duration of 10 min, leading to over 4 M samples per run. These measurements were also conducted outside our institute. Due to the number of samples obtained per run and to construct valid confidence intervals, we apply the batch means method [23].

[5]http://www.comsys.rwth-aachen.de/short/powergraph



(a) Power consumption for different modes. As baseline, the phone is set to Airplane mode, the display illumination is lowered and no other apps than MIRCO are running. Shown are the averages and the 95% confidence intervals.



(b) Results from Section IV-A and the basic power consumption results combined.

Fig. 5. Measurement results for the power consumption of a Nexus 5 in different setups.

Figure 5(a) depicts the results obtained in our measurements. Values shown are the averages of the respective runs. As a baseline, we set the device to *Airplane* mode, set the display illumination to the lowest possible level and no other apps besides MIRCO are running. In this mode, the device consumes 119 mA. For all other measurements, we additionally switch on the respective interface and connect it to the network. Downloading a file over a cellular connection from our signing proxy yields a power consumption of 245 mA for 2G, 312 mA for 3G, and finally 428 mA for a connection via LTE. Afterwards, we set the smartphone back to Airplane mode but activate the Wi-Fi interface to connect to the mobile hotspot operated by the other smartphone. Again, the device starts downloading, which results in 423 mA. Moreover, we measure the power consumption of opening a hotspot and sending a file, producing a power consumption of 549 mA.

Finally, we connect the smartphone to the cellular network and to the mobile hotspot. The device does not send data via the cellular connection but downloads content from the other smartphone via its Wi-Fi interface. We argue that this reflects the actual usage, as we believe that smartphone users do not explicitly deactivate their cellular interface when not needed. The results show that the power consumption is mainly dominated by the Wi-Fi interface and that the idling cellular interface does not increase the power consumption significantly. Furthermore, these results indicate the impact a malicious user could have on other clients by forcing them to download or send content unnecessarily, with respect to energy consumption.

In the next step, we incorporate the power consumption measurements into our timing evaluation. Therefore, we extract the fastest observed goodput rate for all connection modes during the week and the weekend and calculate the duration it would take to transfer 1 MB of payload. We combine this duration with the power consumption in each mode respectively and obtain the results presented in Figure 5(b). Please note that

for the D2D transfer rates using Wi-Fi, the rates during the week and the weekend are stable and therefore not separated here. Thus, also from an energy point of view, the absence or unavailability of 3G and LTE show the benefits of mobile content offloading in comparison to 2G as it performs at least one order of magnitude worse than using a local D2D link. Moreover, the local link is also more energy efficient than 3G and performs similarly to LTE. In typical mobile content offloading systems, malicious participants could still fool others into downloading and wasting energy, however by design, MIRCO excludes these participants based on reputation.

Our results underline the importance of detecting abuse in mobile content offloading. Specifically, such detection serves to stop (or even prevent by discouragement) the misuse and waste of communication and energy resources. The severity of this misuse and therefore the potential savings when excluding misbehaving clients, is for example illustrated by the amount of energy necessary for the local exchange of content via Wi-Fi, as shown in Figure 5(a) (*Wi-Fi rx* and *tx*). Here, MIRCO offers a tool to detect and report misuse and subsequently identify and exclude the respective clients in order to sustain the cooperative nature of mobile offloading.

V. Conclusion

In this paper, we address the missing support for content authenticity and participation reputation in mobile content offloading. We propose MIRCO, adding descriptive information and cryptographic signatures to the initial local caching of content on mobile devices via a signing proxy. Participating devices are then able to verify the authenticity of content in further offloading interactions. Moreover, MIRCO introduces client authentication and a cryptographic commitment to D2D content exchanges, allowing to report the success or failure to a public reporting system that traces the reputation of individual clients. The detection of abuse can then be countered with the revocation of client credentials and, in effect, the exclusion of the respective client from future offloading interaction.

Our implementation on commodity Android smartphones shows the real-world applicability of MIRCO. In comparison with cellular communication, MIRCO introduces only negligible time overhead within the D2D content exchange and benefits from the energy efficiency of local D2D communication. Future work targets a real-life deployment of MIRCO. Moreover, we plan on extending the reporting system in order to preserve user privacy with respect to content linkability.

Acknowledgments

This work has been funded by the German Research Foundation (DFG) as part of project A03 within the Collaborative Research Center (SFB) 1053 “MAKI – Multi-Mechanism-Adaptation for the Future Internet”.

References

[1] Cisco, San Jose, CA, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014-2019,” *Cisco Public Information*, 2015.

[2] C. Shi, K. Joshi, R. K. Panta, M. H. Ammar, and E. W. Zegura, “Coast: Collaborative application-aware scheduling of last-mile cellular traffic,” in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’14. New York, NY, USA: ACM, 2014.

[3] J. Erman, A. Gerber, M. Hajiaghayi, D. Pei, S. Sen, and O. Spatscheck, “To cache or not to cache: The 3g case,” *Internet Computing, IEEE*, vol. 15, no. 2, March 2011.

[4] A. Finamore, M. Mellia, Z. Gilani, K. Papagiannaki, V. Erramilli, and Y. Grunenberger, “Is there a case for mobile phone content pre-staging?” in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT ’13. New York, NY, USA: ACM, 2013.

[5] B. D. Higgins, J. Flinn, T. J. Giuli, B. Noble, C. Peplin, and D. Watson, “Informed mobile prefetching,” in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’12. New York, NY, USA: ACM, 2012.

[6] J. Jiang, S. Zhang, B. Li, and B. Li, “Maximized cellular traffic offloading via device-to-device content sharing,” *Selected Areas in Communications, IEEE Journal on*, vol. PP, no. 99, 2015.

[7] H. Wirtz, J. Rüth, T. Zimmermann, and K. Wehrle, “Interest-based cloud-facilitated opportunistic networking,” in *Proceedings of the 8th ACM MobiCom Workshop on Challenged Networks*, ser. CHANTS ’13. New York, NY, USA: ACM, 2013.

[8] X. Bao, Y. Lin, U. Lee, I. Rimac, and R. R. Choudhury, “DataSpotting: Exploiting naturally clustered mobile devices to offload cellular traffic,” in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013.

[9] Y. Li, G. Su, P. Hui, D. Jin, L. Su, and L. Zeng, “Multiple mobile data offloading through delay tolerant networks,” in *Proceedings of the 6th ACM workshop on Challenged networks*. ACM, 2011.

[10] F. Rebecchi, M. Dias de Amorim, and V. Conan, “Flooding data in a cell: Is cellular multicast better than device-to-device communications?” in *Proceedings of the 9th ACM MobiCom workshop on Challenged networks*. ACM, 2014.

[11] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, “Cellular traffic offloading through opportunistic communications: A case study,” in *Proceedings of the 5th ACM Workshop on Challenged Networks*, ser. CHANTS ’10. New York, NY, USA: ACM, 2010.

[12] M. Pitkänen, T. Kärkkäinen, J. Greifenberg, and J. Ott, “Searching for content in mobile dtns,” in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*. IEEE, 2009.

[13] H. Wirtz, M. Ceriotti, B. Grap, and K. Wehrle, “Pervasive Content-centric Wireless Networking,” in *15th Symposium on A World of Wireless, Mobile and Multimedia Networks*, ser. WoWMoM ’14, 2014.

[14] F. Qian, K. S. Quah, J. Huang, J. Erman, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, “Web caching on smartphones: ideal vs. reality,” in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012.

[15] S. Trifunovic, C. Anastasiades, B. Distl, and F. Legendre, “PodNetSec: Secure opportunistic content dissemination,” in *ACM Mobisys Demo Session*, 2010.

[16] J. Scott, J. Crowcroft, P. Hui, and C. Diot, “Haggle: A networking architecture designed around mobile users,” in *WONS : Third Annual Conference on Wireless On-demand Network Systems and Services*, 2006.

[17] R. Bruno, A. Masaracchia, and A. Passarella, “Offloading through opportunistic networks with dynamic content requests,” in *Mobile Ad Hoc and Sensor Systems (MASS), 2014 IEEE 11th International Conference on*. IEEE, 2014.

[18] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, P. D. Gallagher *et al.*, “NIST Special Publication 800-57 Recommendation for Key Management, Part 1, Rev 3: General,” *NIST Special Publication*, 2012.

[19] T. Berners-Lee, R. Fielding, and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” RFC 3986 (INTERNET STANDARD), Internet Engineering Task Force, Jan. 2005, updated by RFCs 6874, 7320. [Online]. Available: http://www.ietf.org/rfc/rfc3986.txt

[20] S. Trifunovic, B. Distl, D. Schatzmann, and F. Legendre, “WiFi-Opp: ad-hoc-less opportunistic networking,” in *CHANTS*. ACM, 2011.

[21] P. Leach, M. Mealling, and R. Salz, “A Universally Unique IDentifier (UUID) URN Namespace,” RFC 4122 (Proposed Standard), Internet Engineering Task Force, Jul. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4122.txt

[22] D. Eastlake, “Transport Layer Security (TLS) Extensions: Extension Definitions,” RFC 6066 (Proposed Standard), Internet Engineering Task Force, Jan. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6066.txt

[23] A. M. Law, *Simulation modeling and analysis*. McGraw-Hill, 2007.