This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

An Anonymous End-to-End Communication Protocol for Mobile Cloud Environments

Claudio A. Ardagna, Mauro Conti, Mario Leone, and Julinda Stefa

Abstract—The increasing spread of mobile cloud computing paradigm is changing the traditional mobile communication infrastructure. Today, smartphones can rely on virtual (software) "clones" in the cloud, offering backup/recovery solutions as well as the possibility to offload computations. As a result, clones increase the communication and computation capabilities of smartphones, making their limited batteries last longer. Unfortunately, mobile cloud introduces new privacy risks, since personal information of the communicating users is distributed among several parties (e.g., cellular network operator, cloud provider). In this paper, we propose a solution implementing an end-to-end anonymous communication protocol between two users in the network, which leverages properties of social networks and ad hoc wireless networks. We consider an adversary model where each party observing a portion of the communication possibly colludes with others to uncover the identity of communicating users. We then extensively analyze the security of our protocol and the anonymity preserved against the above adversaries. Most importantly, we assess the performance of our solution by comparing it to Tor on a real testbed of 36 smartphones and relative clones running on Amazon EC2 platform.

Index Terms—Anonymity, Mobile cloud computing, Mobile communications, Smartphone clones

1 Introduction

Recent advances of mobile technologies have turned our smartphones into small, powerful devices that we use not only to call and text, but also to play, check our emails, watch movies, wherever and whenever we are. Current smartphones in fact come with built-in 3G/Wi-Fi and Bluetooth technologies, and their processors outperform those of the desktop computers of ten years ago. Yet, current devices suffer from a major drawback: their battery limit. Indeed, current batteries cannot cope with the ever increasing complexity of mobile applications, that become more and more energy hungry. To address the need of optimizing energy consumption [1], some solutions focus on offloading mobile computation to software clones of real devices in the cloud [2], [3], [4], [5], [6], [7], [8]. Advancing in this line, platforms like *Clone2Clone* (C2C) [4], [9], [10], where clones are peer-to-peer connected to each other in the cloud, allow for computation and communication offloading.

The above scenario involves many entities including the devices, the clones, the cloud provider, and the cellular network operator. In particular, the last two handle all communications, and can monitor (and possibly, eavesdrop) within the respective network: who is communicating with whom, how often, and what amount of data is being exchanged. If they collude, all this information can be easily inferred for end-to-end communications, thus threatening the privacy of all users in the system. The mobile cloud computing scenario is therefore exacerbating the mobile privacy problem [11], which turns the risk of implicit total surveillance of individuals even more into a reality.

In this paper, we focus on the above privacy problem and propose a communication protocol that allows smartphone users to anonymously communicate in a mobile cloud computing environment. To this aim, the sender of a communication involves other smartphones, clones, and the cellular operator in the communication towards the receiver. We assume communications that complete in few seconds, since they characterize many of everyday mobile communication sessions [12]. Our solution relies on opportunistic ad hoc communications between smartphones and on social-network properties to provide anonymity. We design our scheme assuming all parties eavesdropping on the communication as potential adversaries that possibly collude among them.

Our solution offers features that would *not* be available using Tor [13]. First, the authors of Tor [13] clearly state that providing resiliency against end-to-end attack (e.g., end-to-end time and size correlations) is a non-goal for Tor, while it is mandatory in our scenario. Furthermore, Tor assumes an adversary observing only a fraction of the network traffic. In our scenario, where communication patterns always involve the clones of the sender and receiver devices, this assumption does not hold. As an example, the cloud provider, which eavesdrops on all communications in its domain, can act as a global attacker in the communication between the clones of the sender and receiver devices. Moreover, our solution relies on a multi-hop ad hoc wireless local communication that hides the initiator of the communication to the cellular operator, that is, the sender of our anonymous communication protocol. This feature would not be available if the sender device directly asks the cellular operator to

• *C.A. Ardagna is with the Dipartimento di Informatica, Università degli Studi di Milano, Crema–Italy; E-mail: claudio.ardagna@unimi.it. M. Conti and M. Leone are with the Dipartimento di Matematica, Università di Padova, Padova–Italy; E-mail: conti@math.unipd.it, mario.leone.23@gmail.com. J. Stefa is with the Dipartimento di Informatica, Sapienza Università di Roma, Roma–Italy; E-mail: stefa@di.uniroma.it.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

contact a Tor Router. Finally, Tor relies on a significant amount of traffic being present in the network to provide anonymity, while this is not needed in our solution.

The contribution of the proposed solution, which considers the novel aspect of anonymity in mobile-cloud integrated infrastructures, is twofold: *i)* it provides a complete end-to-end anonymous channel, allowing users to anonymize their communication profiles against adversaries including the cellular operator and the cloud provider; *ii)* it allows low-cost and battery-preserving anonymous communications. The work in [14] provided a preliminary version of the solution presented in the current paper. In particular, this paper extends the work in [14] in several directions as follows. First, it refines the anonymous end-to-end communication protocol clarifying the communication flow and the role of the proxy (Section 4). Second, it proposes a detailed security analysis (Section 5) evaluating: *i)* the level of anonymity against single and colluding adversaries eavesdropping on the communication; *ii)* the level of anonymity against timing and predecessor attacks; and *iii)* the impact of our solution on users privacy. Most importantly, we implement our anonymity protocol on a real testbed of 36 Android powered devices and assess its performance in terms of system responsiveness and energy consumption in comparison with Tor for mobile Android systems (Section 6). Our experimental results (Section 7) show that our solution outperforms Tor in terms of energy consumption, while being comparable to it in terms of time overhead.

2 Related Work

Achieving anonymous communications is an important and well studied issue in both wired and wireless systems. So far, the most applicable schemes are based on the concept of mixing [15], where messages are sent along a chain of proxy nodes (called mixes) that accumulate and forward source-encrypted messages in batches. Tor [13], perhaps the most popular deployed mix network in wired systems, achieves mixing by layer-encrypting a message at the source, and decrypting it once at each hop of a source-selected chain of proxy relays (called also Tor nodes). The last Tor node of the chain sends the unencrypted message to the destination specified by the client. Our solution leverages for some communication steps an approach similar to the one of mix zones, while for some steps the unpredictability of the originator of a communication is basically due to the hidden terminal property of wireless communications.

The enormous popularity of social network platforms has given also raise to new computing and anonymity schemes that rely on friendship relations among users [16], [17], [18], [19], [20]. By assuming that friend nodes trust each other, schemes like [16], [17], among others, show how anonymity mechanisms based on social-trust offer the same anonymity properties of Tor, yet lowering the delay of the communication and alleviating the computation burden of the source. The work in [18] relies on trust relationships in social networks to build circuits for onion routing. These circuits are at the basis of a decentralized approach for anonymous communications. The work in [19] considers the anonymity problem in social networks, and formally evaluates the role of dynamic social graphs in the context of anonymous communication systems. The work in [20] investigates the SocialCloud computing paradigm, where computing nodes have trust relationships and are bound by social ties. Our solution also leverages trust among friend nodes to help anonymizing the communications.

Early approaches proposing anonymous communication schemes for wireless systems are mostly inspired by Tor-like networks [21], [22], [23]. They either rely on source-routing, or assume a reliable network where full connectivity among peers is available. These requirements make them inapplicable in highly dynamic mobile wireless networks like the Pocket Switched Networks (PSNs). In PSNs, users carry around devices communicating with short-range technology (Bluetooth or Wi-Fi), and the communication links appear and disappear over time, as the device holders get in physical proximity. To the best of our knowledge, the solutions in [24], [25], [26] are the only approaches that cope with the intermittent nature of PSNs. All these schemes envision scenarios where nodes communicate in multiple hops, without a fixed networking infrastructure, and rely on the social-trust: "friends" rely on their "friends" in the network to achieve the required anonymity.

More recently, the paper in [27] presents an anonymous communication protocol aimed to preserve (α,β)-anonymity in mobile hybrid networks, involving cellular and Wi-Fi communication links. Although this approach provides communication anonymity in a scenario where mobile users move and form networks of arbitrary topology, it is not applicable to our settings, where the smartphones constantly communicate with their software clones in the cloud to either offload computation [2], [3], [4], [7], [28] or backup/store data for reliability reasons. Much in line with [4], we assume a system where each smartphone is associated with its clone in the cloud, and clones of "friend" users (according to an on line social network, e.g., Facebook) are inter-connected by P2P links between them.

3 System and Anonymity Models

This section presents our system and anonymity models. Table 1 summarizes the notation used in this work.

3.1 System Model

Our system includes four different parties: *i)* the smartphone users, carrying a mobile device for communication and managing a clone of their smartphone in the cloud; *ii)* the cellular network operator, managing the cellular infrastructure and allowing smartphone users to access its services; *iii)* a set of proxies, mediating requests from a smartphone user to a clone in the cloud; and *iv)* the cloud provider, managing the cloud infrastructure and its computing resources. Figure 1 presents our system model. Communications among the entities are denoted with black arrows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

TABLE 1
Notation

D	Set of mobile devices
C	Set of device clones
d_i	i-th mobile device
c_i	i-th clone
S_{d_i}	Friend-set of d_i, (including d_i itself)
S_{c_i}	Friend-set of c_i, (including c_i itself)
s	Sender of a communication
r	Receiver of a communication
o	Cellular network operator
cp	Cloud provider
pr	Proxy
Adv	Adversary attacking users anonymity
SK	Symmetric key
(K^p,K^s)	A public/private key pair
$[m]_K$	Message m encrypted with key K

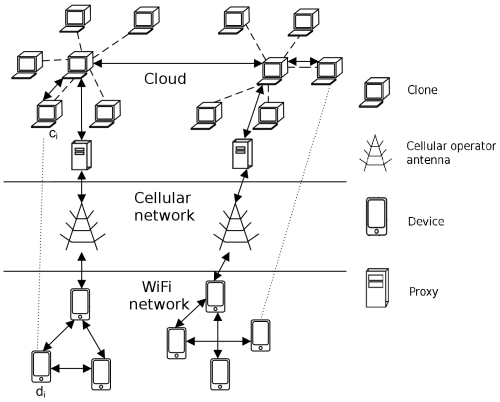


Fig. 1. System model.

The smartphone users communicate with each other through both the cellular network infrastructure (mid-layer in Figure 1) and short-range Wi-Fi ad hoc wireless communication links (bottom layer in Figure 1). We denote with D the set of mobile devices (and with C the set of clones). Each device $d_i \in D$ is associated (dotted lines in Figure 1) with a software clone $c_i \in C$ in the cloud platform, for offloading computations and communications, and for backup purposes. Slightly abusing the notation, in the following when clear from the context, we refer to d_i and c_i also as the *user* of d_i and c_i, and vice versa. In addition, o and cp denote respectively the cellular network operator and the cloud provider. We assume a single cellular network operator o receiving all cellular communications, and acting as a gateway between mobile peers and the cloud infrastructure. Also, we assume the presence of a single cloud provider cp managing and observing all cloud communications. While phones can connect to the cloud via other means beyond the cellular operator (e.g., through WiFi), in this work we consider only the cellular network, as being the connection available to smartphones most of the time. It is important to notice that from the architecture point of view, our solution works independently from the type of connection provider. Furthermore, the better the available data connection, the better the performance of our protocol.

The clones are inter-networked among them through P2P links in the cloud. In particular, some of these links (dashed lines in Figure 1) reflect the friendship relations among the respective users, and define well connected social-communities of clones in the cloud. These links can be either declared by the users (as in Facebook), bootstrapped by existing online social networks, or a mix of the two. We denote with S_{d_i} (S_{c_i}) the set of devices (clones) whose users are friends with the user of device d_i (clone c_i). By definition, S_{d_i} and S_{c_i} are of the same size. Also, we note that a user might belong to several, and possibly overlapping, sets of friends (e.g., those defined by her gym friends and her school friends). As in [16], [17], we assume that $|S_{d_i}|>1$ and that there is a trust relationship among friend-users. The same relationship holds also among the respective clones in the cloud. We assume that the friendship information of users in the system is public and accessible through a centralized friendship database (e.g., the public friendship information available in Facebook). In addition, we assume that each device d_i shares a symmetric secret key SK_{d_i} with its clone c_i, which is used to preserve confidentiality in the communication between them. Moreover, each clone c_i is provided with a public/private key pair ($K^p_{c_i}$,$K^s_{c_i}$). We will denote with $[m]_K$ a message m encrypted with key K.

Our system architecture also includes a set of n proxies $pr_1, \ldots, pr_n$, which mediate all communication channels between the devices and the clones in the cloud. Proxies keep a map between clone ID (which is also available in the friendship database) and clone public IP address. As we will discuss in Section 5, the role of the proxies is to decouple cellular and cloud identities, making successful attacks to the anonymity of our protocol impervious to colluding o and cp. Each proxy pr_i is provided with a public/private key pair ($K^p_{pr_i}$,$K^s_{pr_i}$) used when communicating with the user's device or clone. For simplicity, in the following, we consider a single proxy pr.

Finally, we consider each communication among devices as a (bi-directional) exchange of a set of messages (black arrows in Figure 1) between the communication sender $s \in D$ and the communication receiver $r \in D$. We note that the actual receiver of a communication originated by s could be the clone c_s of s, the device d_r of r, or a clone c_r. The latter case models a scenario where a server (e.g., providing a cooperative sensing application), which would like to join our protocol, implements a clone acting as the interface with the real server application.

3.2 Anonymity Model

Extending the concept of k–anonymity [29],[1] we aim to hide the sender of the communication among (at least) α possible devices, as well as hide the receiver among (at least) β possible devices, with α and β chosen by the sender. Formally, we use the following definition.

Definition 3.1 ((α,β)–anonymity): Given a sender s and a receiver r, a communication between them is (α,β)–anonymous, if an adversary Adv cannot associate device s to less than α devices, and device r to less than β devices.

1. A release of data is k–anonymous if it can be indistinctly matched to at least k respondents [29].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

A communication is (α, β)–anonymous against Adv, if Adv cannot identify s as the sender with probability higher than $1/\alpha$ and r as the receiver with probability higher than $1/\beta$. When Adv has no information on the identity of s (r, resp.), α (β, resp.) assumes value $*$, meaning that the probability of identifying the sender (receiver, resp.) of a communication tends to zero. We note that $\alpha{=}*$ and $\beta{=}*$ represents the strongest requirements on the sender and receiver anonymity, while $(*,*)$–anonymity the strongest requirement on communication anonymity. We also note that the general case is for $\alpha{\neq}\beta$ and $\alpha, \beta{>}1$, while specific scenarios correspond to different levels of anonymity:

- $\alpha{=}\beta$: s and r are protected at the same level.
- $\alpha{=}1$: s can be identified as being the sender of a communication with one among β possible receivers.
- $\beta{=}1$: r can be identified as being the receiver of a communication with one among α possible senders.
- $\alpha{=}1, \beta{=}1$: s and r are unambiguously identified as the sender and receiver of a communication (no anonymity).

User communications can travel on ad hoc wireless, cellular, or cloud network channels, and are exposed to attacks in each of them. The adversaries in our model are honest but curious, meaning that they neither tamper with the exchanged messages nor can read encrypted messages. They are grouped in four categories as follows.

- *Malicious devices*. They are devices owned by malicious adversaries in the ad hoc wireless network that intentionally attack the anonymity of communicating users.
- *Cellular network operator*. It observes all cellular communications between users, and users and clones. Cellular network operator can measure the position of each device, at least by observing the cellular antenna (cell in the following) the device joins.
- *Cloud provider*. It eavesdrops all the incoming and outgoing traffic for every clone. We assume the cloud provider to do not exploit the cryptographic keys used by the clones, even if they are under its physical control. We believe that solutions employing cryptographic techniques like homomorphic encryption, or hardware-based protections like tamper-proof USB-token (provided by the client and accessed by the clone on the cloud) could keep the keys safe from a physical access by the cloud provider. However, since a proper design and description of such solutions is not trivial, due to space limitations, we left them for our future work.
- *Malicious clones*. They are clones under the control of malicious devices that intentionally attack the anonymity of communicating users.

Adversaries can collude among them to identify source s and destination r of the communication or, in other words, to reduce the anonymity in Definition 3.1 to (1,1)–anonymity. We note that the proxy is trusted and does not collude with any of the adversary, although, as will discuss further in Section 5, our solution is resilient to the scenario in which it is compromised by malicious adversaries. In addition, a device or clone does not attack or collude with an adversary to compromise the anonymity of a friend device or clone. Finally, we assume the clone of the sender and of the receiver to be not compromised by an attacker.

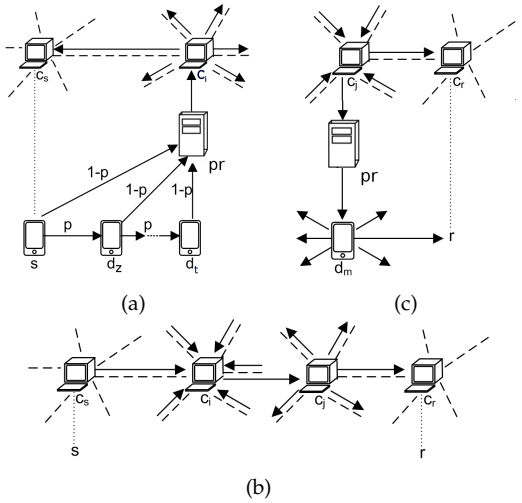


Fig. 2. Communication steps (a) sender communication, (b) clone communication, (c) receiver communication.

4 Our Solution

The main goal of our protocol is to achieve (α, β)–anonymity for communications involving a sender s and a receiver r, with part of the communications handled by their clones c_s and c_r in the cloud. In the protocol, s and r communicate as follows. First, s anonymously sends a message to its clone c_s (sender communication). This is achieved through a probabilistic multi–hop Wi-Fi forward to devices in the physical proximity of s and a forward in the cloud through a proxy and a set of clones. We note that WiFi devices are in physical proximity when they can directly communicate with no support by fixed equipments (e.g., access points) or other devices. Second, upon reaching c_s, the message is anonymously forwarded to clone c_r of r (clone communication). This involves a clone c_i in the same social network with c_s, a clone c_j in the same social network with c_r, and a subset of friend clones of c_i and c_j. Third, upon reaching c_r, the message is (possibly) distributed to mobile device r, via a proxy, the cellular operator, and a device in the proximity of r (receiver communication). Finally, r (or directly its clone) can reply to s by either re-using the same path or by building a new one (response communication).

Figure 2 shows the distribution of messages in the above steps among parties, whereas Figure 3 depicts the communication flow in our scheme illustrating the content of each message. The details of these steps are discussed in the remaining of this section. To this aim, we consider s defining anonymity parameters α and β, and sending a message with payload m to r.

Sender Communication. User s looks up in the friendship database and randomly selects one of her friend clones c_i whose social network (S_{c_i}) has at least α members, that is, $|S_{c_i}| \geq \alpha$. Then, s sends the message to proxy pr, using a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

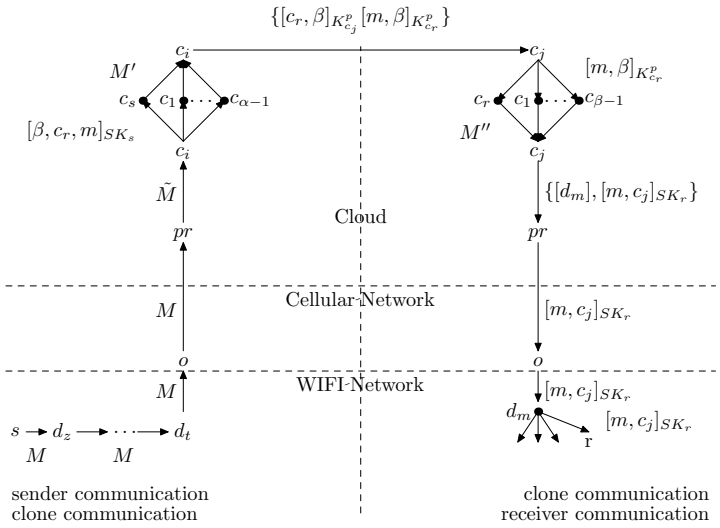


Fig. 3. Communication flow for our protocol.

probabilistic multi–hop Wi-Fi forward to devices in its physical proximity. Upon receiving the message, pr forwards the message to c_i. At this stage c_i delivers the message to α of its friends including c_s, which is the only one able to decrypt the message.

To this aim, user s prepares a bundle M that contains *(a)* the identity of c_i encrypted by K^p_{pr} (the public key of pr), *(b)* the identity of c_s and parameter α encrypted with $K^p_{c_i}$ (the public key of c_i); *(c)* parameter β, the identity of clone c_r and payload m encrypted with SK_s (the shared secret key between s and c_s), as follows:

$$M = \{[c_i]_{K^p_{pr}}, [c_s, \alpha]_{K^p_{c_i}}, [\beta, c_r, m]_{SK_s}\}. \quad (1)$$

After producing M, s checks the number of devices in its proximity, reachable by ad hoc Wi-Fi links. This can be achieved by having devices periodically broadcast *Probe request* through the Wi-Fi network, to announce their presence to neighbors in their proximity. User s waits till it is "surrounded" by at least $\alpha-1$ other devices, before sending M. This step avoids a device in user's proximity from guessing with probability higher than $1/\alpha$ the identity of s. User s then sends M with probability p to a device randomly selected among the nearby devices (e.g., device d_z in Figure 2(a)), or with probability $1-p$ directly to pr through cellular operator o. Each receiving device applies the same approach until message M is sent to proxy pr (e.g., d_t in Figure 2(a)). Intuitively, the probabilistic message forward aims to confuse a curious cellular network operator, which might leverage the user identifier (e.g., the SIM number) and information on the position of forwarding device d_t to guess the identity of s, by implementing a path with variable length. At each step in fact the peer probabilistically decides to either forward the message to another peer or directly sends it to the proxy, assuming no failure in the communication between peers.

When proxy pr receives bundle M, it retrieves the identity of c_i using its private key K^s_{pr} and forward the rest of the bundle to c_i:

$$\tilde{M} = \{[c_s, \alpha]_{K^p_{c_i}}, [\beta, c_r, m]_{SK_s}\}. \quad (2)$$

We note that pr can neither access the identity of c_s nor of c_r. Upon clone c_i gets $\tilde{M}$, it retrieves the identity of c_s and parameter α by decrypting $[c_s, \alpha]_{K^p_{c_i}}$ with its private key $K^s_{c_i}$. Then, it forwards the remaining of the bundle, $[\beta, c_r, m]_{SK_s}$, to a subset of α clones in its social network including c_s. Among the α clones receiving the message, only c_s is able to decrypt $[\beta, c_r, m]_{SK_s}$ using SK_s.

Clone Communication. Each of the α clones receiving $[\beta, c_r, m]_{SK_s}$ replies, after a timespan τ, with a response to c_i (see Figure 2(b)). For c_s, the response is a bundle M' which will be then forwarded towards destination clone c_r. The responses of the other clones are randomly generated, have variable lengths, and are encrypted with c_i's public key. We note that τ is chosen so that it allows c_s to compute the response message. In this way, cloud provider cp cannot tell, among the α replying clones, which one is the actual source clone c_s.

Let us get back to M' contained in the response of c_s sent to clone c_i. To generate it, clone c_s looks up in the friendship database and picks, from the friend set of c_r, a clone c_j with more than β friends. Then, it prepares a bundle M' containing *(a)* the identity of c_j, *(b)* the identity of clone c_r and parameter β encrypted with $K^p_{c_j}$ (the public key of c_j), *(c)* payload m and parameter β encrypted with $K^p_{c_r}$ (the public key of c_r). The resulting message M' is then encrypted with the public key of clone c_i and sent back to c_i, after timespan τ elapses. M' looks as follows:

$$M' = \{c_j, [c_r, \beta]_{K^p_{c_j}}, [m, \beta]_{K^p_{c_r}}\}_{K^p_{c_i}}. \quad (3)$$

Upon receiving M' from c_s, c_i decrypts it, retrieves c_j, and forwards message $\{[c_r, \beta]_{K^p_{c_j}}, [m, \beta]_{K^p_{c_r}}\}$ to c_j. The responses of the other clones are simply dropped. Upon receiving $\{[c_r, \beta]_{K^p_{c_j}}, [m, \beta]_{K^p_{c_r}}\}$, c_j decrypts $[c_r, \beta]_{K^p_{c_j}}$ using its private key $K^s_{c_j}$. Then, c_j forwards $[m, \beta]_{K^p_{c_r}}$ to a subset of β clones in its social network including c_r. Among clones receiving the message, only c_r is able to decrypt $[m, \beta]_{K^p_{c_r}}$ using its key $K^s_{c_r}$.

Receiver Communication. We propose a solution that follows a *push* approach to deliver payload m to receiver r (see Figure 2(c)). First, similarly to the previous phase, all β clones that received a message from c_j, after a timespan τ, reply to it with a response. The response of c_r is a bundle M'' that includes the message to be sent to r. M'' contains *(a)* the identity of a device d_m in the proximity of r (reachable through Wi-Fi), *(b)* payload m and the identity of c_j encrypted with SK_r (the secret key of r), as follows:

$$M'' = \{[d_m], [m, c_j]_{SK_r}\}_{K^p_{c_j}}. \quad (4)$$

M'' is encrypted with the public key of clone c_j and sent to it as a response, after timespan τ elapses. As in the previous case, since β responses are sent to c_j from β different clones, cp cannot tell which is the one generated by the destination clone, and containing the actual message for the real device.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

We note that clone c_r might be the actual destination of the message. In this case, c_r will inform c_j that the procedure should stop here, by adding a special string (e.g.,"stop") instead of d_m in M'' (see Equation 4). We observe that in this case, the other $\beta-1$ clones do not need to have the knowledge of whether c_r is the destination or not. Otherwise, the protocol proceeds with the remaining part of the *receiver communication* step as follows.

Upon receiving M'', c_j sends $\{[d_m], [m, c_j]_{SK_r}\}$ to pr, which in turn retrieves the identity of d_m and forwards $[m, c_j]_{SK_r}$ to it. Then, d_m broadcasts the received message to the nearby devices. Among other devices, r receives the broadcasted message, decrypts it with SK_r, and reads the payload. We note that, for this to work, c_r must know which devices are currently in physical proximity of r. To this aim, r periodically notifies c_r of neighboring devices around it, that is, the ones from which it receives a *Probe request* including the number of their neighboring devices. In fact, neighboring devices with less than β devices in their proximity would expose the anonymity of r, if selected as destination d_m. Clone c_r selects one of the neighbors of r with at least β neighboring devices. If this privacy condition is not met, then c_r would simply ask c_j to stop the procedure (as described above for the scenario where c_r is the final destination). This communication, called *neighbor notification*, is the only one external to our protocol.

Response Communication. If the communication between s and r is bi-directional, two approaches are possible: *i)* the protocol is repeated as a one-way communication switching s with r; *ii)* the message from r to s follows the same path used by the message received by r. In *ii)*, involved clones c_i and c_j must be the same. To this aim, message M'' contains the identity of c_j, and c_j must keep track of c_i in the sender (now receiver) social network. Note that, if one sender communicates frequently with a particular receiver, an adversary may be able to identify this pattern by correlating and intersecting the different sets of α and β supporting clones selected in the social network of c_i and c_j, respectively, among different communications. To avoid this, c_i and c_j must choose the same sets of α and β clones.

To conclude, for preference α=1, sender s does not involve Wi-Fi neighbors in its proximity during the sender communication phase, while it directly sends M to pr. For preference β=1, clone c_r directly sends message M'' to r via pr, bypassing d_m in the receiver communication phase.

5 Security Analysis

We first analyze the impact of eavesdropping attacks against (α, β)–anonymity provided by our protocol. In particular, we present a security analysis against: single adversaries (Section 5.1), colluding adversaries belonging to the same category (Section 5.2), and colluding adversaries belonging to different categories (Section 5.3). We then evaluate the robustness of our solution against timing and predecessor attacks (Section 5.4). We finally discuss privacy issues introduced by our approach (Section 5.5).

5.1 Single Adversaries

Single adversaries are basic adversaries observing a portion of the communication, which build on their own knowledge to breach the anonymity of the users.

Malicious devices. Malicious devices are a threat to the anonymity of communicating users whenever they are in proximity of either the sender or the receiver. Let us consider the sender case first. According to our protocol, the user that initiates the communication forwards message M through an ad hoc link. However, message M is encrypted, thus, a potential malicious device in proximity is not able to read its content. In addition, no identifiable information of s is added in clear to M. Finally, due to the hidden terminal problem that exists in all IEEE 802.11 communications [30], a neighbor observing a device d sending a message cannot assume d to be sender s that started the ad hoc communication, even if the adversary uses directional antennas. As a consequence, (*,*)-anonymity is preserved.

Let us then consider a malicious device in receiver's proximity. Thanks to our protocol, this attacker can only observe a message broadcasted by a device d_m with no information about the possible receiver. Also, since s is not under the attacker control, $(*, *)$–anonymity is guaranteed. In the unlikely case in which the attacker observes all devices in d_m's proximity, $(*, \beta)$–anonymity is still guaranteed, because according to our protocol d_m broadcasts a message only if it has at least β neighbors. Note that, even if the malicious device itself is the one who gets to send the message in broadcast, it cannot retrieve more information on the communication since the message is encrypted.

Cellular network operator. Similarly to the previous adversary, the network operator observes each message M sent by a device d towards the cellular network and only retrieves the identity of pr as the destination of the message. In addition, this adversary has information about the position of the devices involved in the communication. We note that location information is coarse-grained, thus making guessing of precise device coordinates uncertain [31]. However, since the message by s is sent through a probabilistic multihop Wi-Fi forward to devices in the physical proximity, $(*, *)$–anonymity is preserved in the worst case. The operator in fact is not able to evaluate if s is in the proximity of the device from which it received the message.

The cellular operator also observes the messages sent from proxy pr to device d_m, physically nearby device r, during the *receiver communication*. We note that the cellular operator cannot access S_{c_j} because pr, from which it receives each message towards d_m, hides the identity of c_j. The operator might then try to reduce the destination anonymity to less than β, leveraging the information on the position of cellular devices. However, r notifies to its clone the devices in its proximity with the number of their neighboring devices. If a device with at least β neighbors is chosen to broadcast the message, $(*, \beta)$–anonymity is guaranteed.

The cellular operator can also build two sets of communication endpoints (one for the sender communication

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

$\{(d_t,pr)\}$ and one for the receiver communication $\{(pr,d_m)\}$) to link sender and receiver messages. According to the above discussion, in the worst case of a single communication in which the cellular operator links a message sent by d_t and the corresponding received by d_m, our solution preserves $(*,\beta)$–anonymity.

Clones c_{Adv}. Adversarial clones are not in a friendship relation with c_s (see assumptions in Section 3.2), and observe part of the communication if selected, by either c_i or c_j, in the protocol. These adversarial clones have access to information in the friendship database, and thus cannot guess with probability higher than $1/|S_{c_i}|$ the identity of c_s, and with probability higher than $1/|S_{c_j}|$ the identity of c_r. We recall that both c_i and c_j involve their set of friends when sending/receiving messages in our protocol. So, in the worst case, we have $(\alpha,*)$–anonymity for clones receiving a message from c_i and $(*,\beta)$–anonymity for clones receiving a message from c_j.

Cloud provider. The cloud provider accesses all cloud communications and observes that: *i)* c_i and c_j send messages to (at least) other α and β clones, respectively, in their social networks; *ii)* c_i and c_j communicate among them. (α,β)–anonymity is then protected against the cloud provider.

Adversary tampering with the proxy. Though we assumed the proxy to be trusted, we do not exclude that it can be compromised by a very strong adversary. As a consequence, the adversary has access to the proxy's private key and can decrypt messages during the *sender communication*. Nonetheless, we note that, in *sender communication*, Adv is able to retrieve c_i and infer the identity of c_s with a probability not higher than $1/|S_{c_i}|$. Similarly, in *receiver communication*, the adversary can retrieve c_j and infer the identity of c_r with a probability not higher than $1/|S_{c_j}|$. In addition, although Adv can access the identity of d_m in *receiver communication*, this does not reveal the real destination. Hence, if a single proxy is assumed, our protocol provides (α,β)–anonymity; otherwise, $(\alpha,*)$–anonymity and $(*,\beta)$–anonymity are guaranteed.

5.2 Single-Category Colluding Adversaries

We consider multiple colluding devices or clones, and analyze the security properties of our scheme. Note that, since a single network operator and a single cloud provider are assumed in our system model, single-category colluding attacks are meaningless in these contexts.

Colluding devices. We now consider devices that collude in the Wi-Fi neighborhood of s and r. As for s, when it sends a message, due to the hidden terminal problem, $(*,*)$–anonymity is preserved because no identifiable information of s is added in clear to the message.

As for r, when a device d in its proximity receives a message, $(*,\gamma)$–anonymity is preserved with: *i)* $\gamma{=}*$ if d is not malicious; *ii)* $\gamma{\geq}\beta$ if d is malicious and the number of non-malicious peers in its proximity is at least β; *iii)* $\gamma{<}\beta$, otherwise. In the latter case, similarly to the solution in [27], the rate of malicious clones can be estimated allowing r to receive the communication if and only if β honest devices are probabilistically in the proximity of d. Moreover, a malicious peer d will try to blindly increase (i.e., without knowing the required β) the probability of being selected by sending a *Probe request* message with a forged number of neighbors. To limit its success, all peers publicizing number of neighbors that is excessive (with respect to other neighbors in their proximity) will be discarded and potentially blacklisted. In addition, clone c_r (responsible for selecting the supporting device d_m) can use contextual and historical information to reduce the probability of selecting a malicious device as d_m. This selection strategy is however outside of the scope of this paper.

Colluding clones. We consider colluding clones in the same social network of c_i or c_j. In general, it is difficult for malicious clones to secretly build an attack network without being identified as malicious by a cloud provider. This would require that malicious clones know each other in advance, and that they are able to communicate either off-band or strictly following our protocol. As discussed for single clones, $(\alpha,*)$–anonymity and $(*,\beta)$–anonymity are preserved in the social network of c_i and c_j, respectively. Also assuming a collusion is possible, anonymity is guaranteed if at least α and β clones are honest in the social network of c_i and c_j, respectively. If this is not true, $(1,*)$–anonymity and $(*,1)$–anonymity can still be achieved in the worst case. A collusion between clones in the social network of c_i and c_j assumes support from the cloud provider and is discussed in Section 5.3.

5.3 Multiple-Category Colluding Adversaries

We consider adversaries of different categories that collude to uncover the communication endpoints.

Devices colluding with the cellular network operator. When the colluding devices are in the proximity of the sender, they can only observe a peer sending a message M, without any information about the peer's identity and any assurance about the fact that the peer is the real sender s. This information mixed with the network operator knowledge does not give any additional information on the sender. Thus, $(*,*)$–anonymity is preserved.

If devices colluding with o are in proximity of the receiver, our protocol guarantees $(*,\gamma)$–anonymity with $\gamma \geq \beta$, iff at least β honest peers are around the device receiving the message.

Devices colluding with the cloud provider or clones. No further improvement can be achieved with respect to the single-category colluding adversaries.

Clones colluding with the cloud provider. Clones collude with the cloud provider by letting it know that they are not the sender/receiver of a message. The end-to-end communication anonymity is guaranteed if at least one of the following conditions hold: *i)* α honest clones in the social network of c_i or *ii)* β honest clones in the social network

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

of c_j are involved in the protocol. Similarly to the solution in [27], α and β can be selected by s taking into account the rate of malicious clones in the network.

Cloud provider colluding with the cellular network operator. First, let us briefly recap the knowledge of the cloud provider and network operator. Cloud provider cp monitors all communications in the cloud and can identify c_i and c_j, access the α and β nodes involved in a communication, and associate them to S_{c_i} and S_{c_j} in the friendship database. Network operator o observes each message sent by a device d towards proxy pr (and vice versa). It can *i)* link a device to its associated clone observing *neighbors notification* performed by devices (as discussed in *receiver communication*); *ii)* know to which network cell the devices are connected. Based on this knowledge, cp and o can associate Wi-Fi devices with respective clones, by implementing an *intersection attack* that would break the anonymity of our solution, as follows. Network operator o observes the distribution of devices within cells, builds a list of devices d that send a message to proxy pr, and creates different sets N_d including the id of devices connected to the same cell of d. Cloud provider cp observes clone c that has been reached by a message via pr and retrieves S_c. If s is the only device in N_d or the only one connected to a cell adjacent to the one d is connected to, such that it is also part of S_c, the sender identity is revealed with high probability. The same discussion holds for r.

In our protocol, an intersection attack can be launched during a *sender communication* or a *receiver communication*. However, it is worth to note that this attack can become extremely expensive and computationally heavy. It can be successful only if the network operator and the cloud provider are able to cope with the uncertainty introduced by the multi-hop ad hoc Wi-Fi communication on the sender side and by the proxy, and to link each message entering with the corresponding message exiting the proxy. This is possible in case of a single communication, but it is exponentially complex in the number of parallel communications and can be further complicated by having the proxy implementing simple techniques, like random delay. As an example, consider n parallel communications originated by different senders, involving m social networks (with $m \leq n$), and l devices contacting pr. To launch the intersection attack the colluding adversaries must: (1) calculate N_d for the l peers that contact proxy pr; (2) identify m social networks S_{c_m}; (3) apply the intersection attack to the Cartesian product of N_d and S_{c_m}. Considering the enormous complexity of this attack, and the uncertainty of its result, it is reasonable to think that it becomes impracticable when it comes to real systems. The complexity is even worse if senders s are not part of corresponding N_d.

Finally, in the unlikely case that the adversary is so strong that manages both to launch the intersection attack and simultaneously tamper with the proxy, we recognize that our scheme fails to provide end-to-end anonymity.

Table 2 summarizes the results of our worst-case analysis. Each cell in Figure 2(a) represents a single adversary, while each cell in Figure 2(b) represents a collusion between adversaries on the corresponding row and column. We also note that in case two symbols are added to a single cell, the first refers to the sender communication phase and the second to the receiver communication phase. A single symbol refers to the whole communication. For example, let us focus on the case where one or more devices (row labeled with D) collude with the cloud operator (column labeled with cp). As we see in the corresponding cell, (α, β)–anonymity is preserved in the sender communication phase, while at most (α, γ)–anonymity is guaranteed in the receiver communication phase. Finally, as for the collusion between the network operator and the clones in the cloud, we consider not realistic that the network operator creates clones in the cloud and then run a "social" attack to have its clones being friends of the communicating peers (and hence being involved in their communications). Anyway, if such attach is run, the outcoming result will be as the one (already shown in Table 2) for the collusion between the network operator and the cloud provider.

TABLE 2
Anonymity against single (a) and colluding (b) eavesdropping attacks. ● denotes (α, β)-anonymity; ◐ denotes (α, γ)-anonymity, with $\gamma < \beta$; ◑ denotes (γ, β)-anonymity, with $\gamma < \alpha$; ○ denotes $(1, 1)$-anonymity; NA denotes a collusion that we assume not possible

	–
d	●
o	●
c_{Adv}	●
cp	●
pr	●

(a)

	cp	C	o	D
D	●	●	● / ◐	● / ◐
o	○	NA	●	
C	○	◑ / ◐		
cp	●			

(b)

5.4 Timing and Predecessor Attacks

We evaluate our anonymity scheme against timing [32] and predecessor attacks [33], which have been defined with wired networks in mind.

A timing attack can be exploited during the Wi-Fi communication in the sender communication phase and the clone-to-clone communication in the clone communication phase. In the first case, the attacker accesses the original message timestamp and compares it with current time to measure the time elapsed from the first forwarding of the message. However, differently from wired networks, there are no stable paths due to the mobility of peers, and the Wi-Fi channel performance is highly variable and depends on the environment conditions, such as, weather conditions, interference with other devices, mobile and physical obstacles. As a consequence, a timing attack is unreliable and not applicable to our scenario. In the second case, the attacker (e.g., the cloud provider) observes the timestamp of all messages exchanged between clones, and focuses on communications between c_i (c_j, respectively) and the α (β) friends, including c_s (c_r), involved in the protocol. The timing attack is not successful also in this case because our protocol mandate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

each of the α (β) friends receiving a message from c_i (c_j) to reply with a message after timespan τ. To further complicate the attack, our solution can be enriched with an implicit "synchronization" between clones on the basis of a common global clock and a window of forwarding. Given that, by protocol definition, all collaborating clones receive the message in the same time window (approximately in the same time instant), our protocol can be adapted to force all clones to send their responses in the subsequent time window after a random delay (i.e., τ).

Predecessor attacks is based on the assumption that attackers receive messages with a higher rate from the communicating parties. This attack does not hold in our scenario because: *i)* each s sends a single message for each communication round and involves other peers that forward the same message; *ii)* each r receives a message without any additional communication is requested; *iii)* clones c_s and c_r forward the same amount of messages of other clones in the anonymity set.

5.5 Privacy

We analyze the impact of privacy attacks that can be built by exploiting private information exchanged in our anonymity protocol. Let us start from Wi-Fi communications in the sender and receiver communication phases. During these phases, each device periodically broadcast *Probe request* through the Wi-Fi network, to announce its presence to neighbors in its proximity. We note that each *Probe request* message includes the number of neighboring devices and a unique identifier allowing receivers to distinguish multiple messages from the same device. This identifier can be a big pseudorandom number digitally signed by a trusted authority and verified by the receiving device using the authority's public key. So, this case does not present privacy leaks.

Also, let us consider our assumption of having the friendship database public and accessible to everyone. Clearly, this assumption is not always be possible/wanted and might led to growing concerns about the privacy of our users. To mitigate such a scenario, the friendship database can be encapsulated in a trusted party (e.g., the friendship database owner), which takes the role of a middleware and provides selective access to friendship information of those users participating in our protocol. We note that our protocol runs unmodified and all lookups to the friendship database are mediated by the trusted party, which receives as input a request for a given user's friend with at least α (β, respectively) friends and returns as output the identity of the clone under the control of the selected friend. The assumption of having a public database has been made and used in our experiments to test the anonymity of our approach against a more powerful adversary accessing the complete set of social networks and friendship relations.

Finally, our scheme requires the sender/receiver to be surrounded by enough auxiliary smartphones. However, we believe that this is not a restriction. In our everyday life we are constantly surrounded by many other people: coworkers (students) when at work (school), family members or neighbors when at home, people going to the same grocery store/bar/cinema and so on. It is very seldom that a user of our system finds herself all alone, not having anyone in her proximity. Even so, the user can either chose to postpone using the system till she is surrounded by enough other people that guarantee her requested anonymity level, or temporarily go for a lower anonymity level. Lastly, to deal with selfish users we could exploit well-known schemes like [34], [35], [36].

6 Implementation

We present our system that implements device, cloud, and proxy components, while using the existing cellular infrastructure. A complete overview of our system and apps is at https://sites.google.com/site/mcloudanonymity/.

6.1 Device

The device component consists of an Android app that enables devices to anonymously communicate according to our protocol by: *i)* seamlessly interconnecting with other devices in proximity through fast Wi-Fi links; and *ii)* handling the device-clone communication along with notifying the clone of the neighbor lists. The app neither requires modifications to the device, nor needs root permissions to run. An overview of the device component is presented in Figure 4(a), while the implementation details are discussed in the remaining of this section.

Ad Hoc Wi-Fi networking. Our app makes use of Wi-Fi Direct for Android: a Wi-Fi standard that builds upon the IEEE 802.11 infrastructure. We use Wi-Fi Direct to realize ad hoc wireless links among devices. The app's module that deals with the ad hoc connectivity is called *WiFiDirectManager* (see Figure 4(a)), and is implemented as a Java task. The working of the device component is as follows. Periodically it performs a traditional Wi-Fi scan to discover possible devices in its proximity. To this aim, the device alternates between a *search state* and a *listen state*. During the search state, the device sends broadcast *Probe Requests*. During the listen state, the device is put in monitoring mode, capturing broadcast *Probe Requests* sent by proximity devices, and replying with *Probe Responses*.

A device s that needs to use our protocol waits until $\alpha-1$ devices are detected, before initiating the communication. Source s launches then a Wi-Fi Direct connection procedure, choosing randomly one of the $\alpha-1$ devices (i.e., d_z) in its proximity. Once the two devices are connected, the *ClientTask* and the *ServerTask* (see Figure 4(a)) are used to exchange messages. Then, the message travels hop-by-hop, following our protocol as described in Section 4. We also note that, according to Wi-Fi Direct architecture, all ad hoc Wi-Fi links, and in turn, all communications, are protected using WPA2 security.

Reporting the current neighbors to the clone. In our solution, the devices need to notify their clones with the list

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

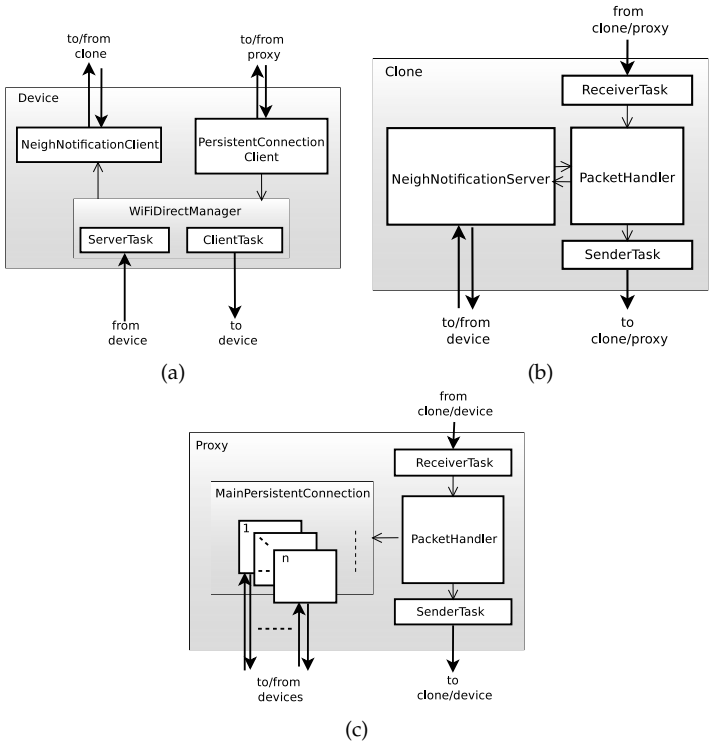


Fig. 4. Architecture of device, clone, and proxy components

of devices in their proximity. This is done any time there is a change in their proximity and is achieved through module *NeighNotificationClient* implemented as a Java thread. The role of *NeighNotificationClient* is to communicate to its counter-part on the clone (described in Section 6.2) the list of MAC addresses of the new devices entering in the proximity, and of those devices leaving the proximity of its device.

Clone-Device communication. A device and its corresponding clone need to communicate to exchange data. Clones are equipped with public IP addresses, which makes the device-clone communication straightforward. On the other side, clones can reach devices only through a "push" mechanism, as devices are not equipped with public IPs. In our system, the push mechanism is implemented through the proxy (described in Section 6.3). On the device side, we implemented module *PersistentConnectionClient*, which: *i)* initiates long-lived TCP/IP connections with the proxy by sending the device's identity based on its MAC address, *ii)* maintains alive the TCP connections over time by occasionally sending keepalive messages or reconnecting in case of loss of connectivity. Reconnections and long keep-alives are based on a timer and exploit the Android OS *AlarmManager* class. As we will discuss in Section 6.3, the proxy maintains a list of devices connected to it, to which it can send push notifications coming from the clones.

6.2 Clone

The cloud side in our implementation is done via software clones of the real devices—virtual Android machines executing on the cloud. Each clone runs as an instance of the Android-x86 system (http://www.android-x86.org/)—the Android OS porting for x86 architectures. We use VirtualBox as our virtualization environment. To enable the clones to run our anonymization protocol, we extend the Android-x86 instance with a series of application-level modules discussed in the following. Figure 4(b) presents an overview of the clone component in our implementation.

Neighbor notification server. Module *NeighNotificationServer* of the clone handles the notifications sent by the associated device about changes of neighbors in device proximity. It is implemented as a persistent Java thread listening to a specific socket, through which it constantly receives notification messages from the device. Whenever a message is received, *NeighNotificationServer* updates its internal proximity buffer accordingly.

Peer-to-peer networking of the clones. Our system is based on the C2C platform [4], in which software clones are interconnected in a peer-to-peer fashion in the cloud. In our clone implementation, peer-to-peer connectivity is achieved through *PacketHandler*—a Java thread responsible for the orchestration of all messages exchanged between clones. It encapsulates the logic to encrypt, decrypt, and verify/validate the integrity of the messages according to our protocol. In particular, during the *receiver communication* phase, module *PacketHandler* communicates with module *NeighNotificationServer* to access information on device d_m in the proximity of receiver r to which it will forward the bundle. Due to the high computation performed by *PacketHandler*, incoming and outgoing traffic is decomposed and handled by threads *ReceiverTask* and *SenderTask*, respectively.

6.3 Proxy

The proxy is implemented in Java, runs on a stand-alone Linux Server, and handles the communications between the devices and the clones in the cloud. Its architectural design is depicted in Figure 4(c), and is composed of four main modules, all implemented as Java threads, described below.

- *MainPersistentConnection* associates each new device d_i entering the system with a new thread named *MainPersistentConnection*$_i$. This thread creates and maintains a persistent TCP connection with the device, which is then used to forward messages coming from the clones.
- *ReceiverTask* is in charge of receiving messages M coming either from the clones or from the devices, and forwarding them to *PacketHandler*.
- *PacketHandler* receives a message M and access its destination. If the destination is a device d_i, the message is forwarded to the respective *MainPersistentComponent*$_i$ thread, which in turn takes care of pushing it to d_i. Otherwise, if the destination is a clone, *PacketHandler* decrypts M using the proxy private key K^s_{pr}, and forward it to *SenderTask*.
- *SenderTask* is responsible of handling messages directed to the clones. Upon a new message arrival, it opens a socket with the destination clone and sends the message to it.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

7 EVALUATION

We show the experimental results obtained using our implementation in a real testbed setup. Towards a full understanding of the benefits of our solution we compared its performance to that of Orbot—an implementation of Tor for Android mobile systems. Orbot provides a local HTTP proxy into the Tor network, hence having the ability to "torify" all of the TCP traffic on an Android device in a transparent way.[2] On top of Orbot we run Gibberbot, a well-known instant messaging client supported by the Guardian Project initiative.[3] In the comparison we focus on energy consumption and system responsiveness (time to achieve end-to-end communications).

7.1 Setup

To assess the performance of our solution we setup two different testbeds. The first (9 smartphones and 9 respective clones) exploited local servers as cloud platform, whereas, in the second and larger one (36 smartphones and respective clones), we deployed the clones on Amazon's EC2 platform. The testbeds are described in details as follows:

Testbed 1: We used 9 Samsung Galaxy Nexus smartphones (Dual-Core 1.2 GHz with 1GB of RAM), running Android 4.2.2, and respective clones and proxy deployed on a private cloud of five Intel Core (TM) 2@1.8GHz servers of 4GB of RAM each. The servers were running GNU Linux Ubuntu 11.04, and the clones were executed on VirtualBox 4.1.12. In particular, 9 clones were distributed over four servers, while the proxy was deployed on the remaining one.

Testbed 2: It consisted in 35 Samsung Galaxy Nexus smartphones (Dual-Core 1.2 GHz with 1GB RAM), 26 of which emulated by 26 Android x86 instances running on top of VirtualBox 4.1.12 on 6 laptops HP Elite Book 8470p Core I5, 8 GB of RAM, and 1 Tablet Nexus 7 (Quad-Core 1.2 GHz, 1GB of RAM). During the emulation we accurately limited, through VirtualBox, CPU and RAM of each instance so to match that of the real Samsung Galaxy Nexus. The clones in the second testbed run on top of the Amazon EC2 cloud platform. We make use of 36 instances of the Android x86 bundle [4] on top of 36 Amazon EC2 T1.Micro Instances (up to 2ECUs). The proxy was also deployed on top of another Amazon EC2 T1.Micro instance running GNU Linux.

To avoid biased results due to other running apps, in both testbeds we disabled the unnecessary phone services and applications. The smartphones used Wi-Fi technology to communicate with nearby devices, and either HSPA or EDGE network to communicate with the clones.

For the evaluation, we studied the performance of the system on real testbeds in dependence of the parameters $p \in [0,1]$ and α and β. Due to lack of space, here we show results relative to three representative values of p, p=0.1, p=0.5, p=0.8, and combinations of α and β taking values in $\{1, 3, 9, 18, 36\}$. Having α=1 and/or β=1 means that we are evaluating either $(1, *)$ or $(*, 1)$–anonymity. In addition, α=1 is a special case implying p=0. We considered two well-known message types: short (SMS) and long (MMS) messages, whose bodies consist respectively of up to 140 bytes and 600 kbytes.

We will present the time to send a message from s to c_s, c_r, and r, the energy consumption of s, r, d_t, and d_m, and the traffic overhead requested by our protocol in the cloud. The experiment was run 100 times and the values were averaged. Due to space restrictions, we only show the results for the larger testbed of 36 phones and respective clones running on Amazon EC2. However, we note that the results are similar to those of the first testbed.

7.2 Time Performance

Figure 5 shows the results on the time overhead introduced by our system: SMS under HSPA connectivity Figures 5(b)–5(d); MMS under HSPA connectivity Figures 5(e)–5(g); MMS under EDGE connectivity Figures 5(h)–5(j). We report the average time (and corresponding standard deviation in errorbars) to send a message from s to c_s, c_r, and r (Figure 5(a)). Since the results of SMS under EDGE connectivity are very similar to SMS under HSPA, and due to strict space constraints we omit them. For comparison, all figures show the average time needed to send the same message using Orbot, which is independent from the parameters of our protocol α, β, and p.

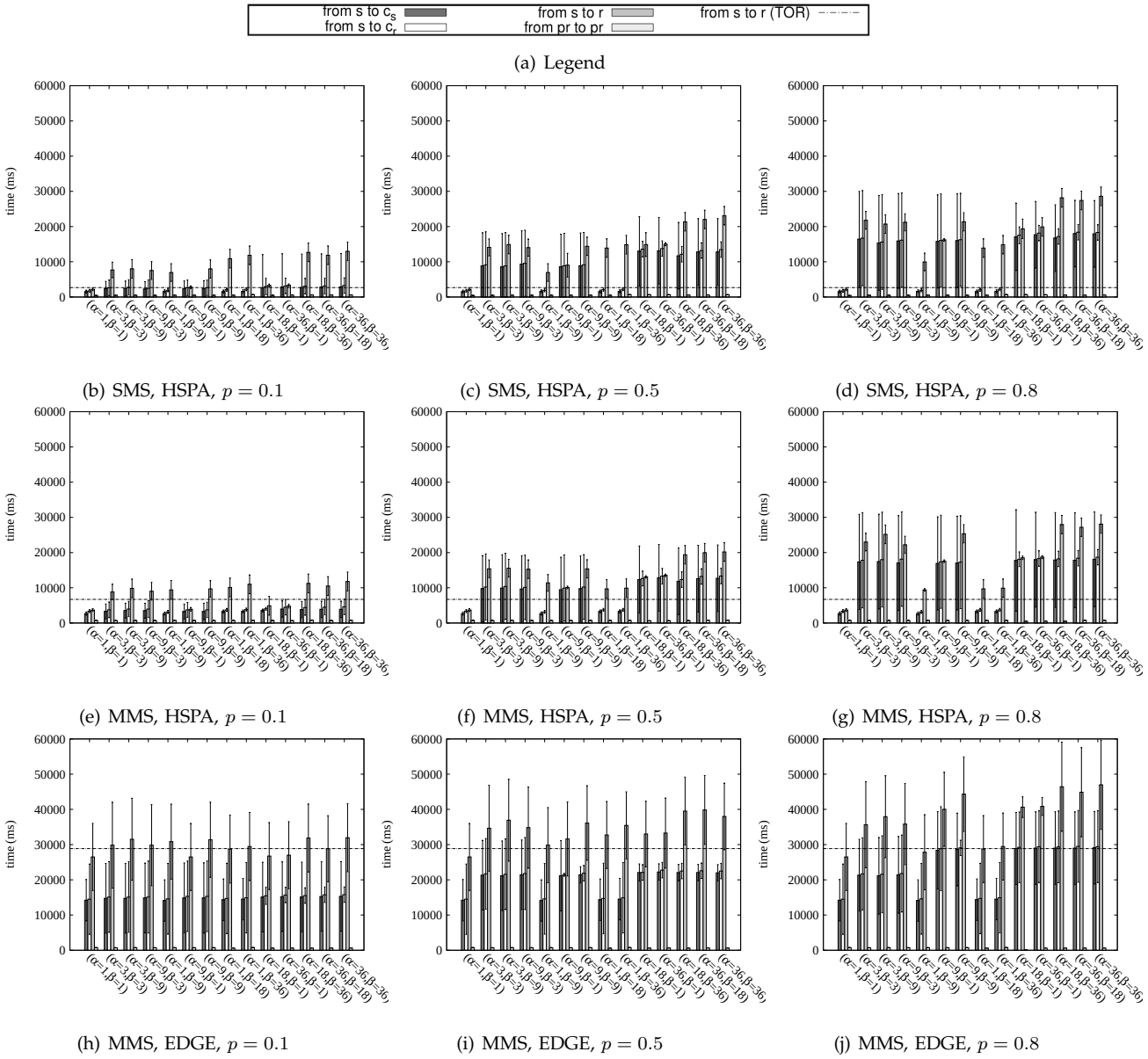
A first important observation is that the Wi-Fi Direct technology is the one that induces large overhead to our system. Each Wi-Fi Direct communication involves negotiation of Wi-Fi Direct group and establishment of WPA2 channels. In the receiver communication protocol this number is determined by β. For large values of β, the overhead induced by Wi-Fi Direct is determining (see Figures 5(b)–5(j)). Differently, on the sender side the number of Wi-Fi Direct communications is independent on α, but determined by p. Thus, for small values of p, the overhead of the Wi-Fi Direct technology is reduced—a high percentage of messages is directly sent by s to pr—whereas it increases for high values of p—devices tend to ad hoc forward messages in their physical proximity. However, we note that the Wi-Fi Direct overhead can be considerably reduced by exploiting future, more performing protocols that allow for seamless ad hoc communication between smartphones.

Another important observation is that the time required to a message received by the proxy during *sender communication* to return to it in the *response communication* is very low, and that the overhead induced by the cloud side for large number of clones (large α and β) does not affect the protocol in a noticeable way (see Figures 5(b)-5(j)). We believe this is a very good result considering that the Amazon instances we used are the less performing, and it can only improve with more powerful instances.

Note also that the cellular technology impacts considerably the performance. For HSPA networking the performance for SMS (Figures 5(b)–5(d)) and MMS (Figures 5(e)–5(g)) are similar, while MMS under EDGE technology performs considerably worse (see Figures 5(h), 5(i), 5(j)). Thus,

2. https://www.torproject.org/docs/android.html.en
3. https://guardianproject.info

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing



(a) Legend

(b) SMS, HSPA, $p = 0.1$

(c) SMS, HSPA, $p = 0.5$

(d) SMS, HSPA, $p = 0.8$

(e) MMS, HSPA, $p = 0.1$

(f) MMS, HSPA, $p = 0.5$

(g) MMS, HSPA, $p = 0.8$

(h) MMS, EDGE, $p = 0.1$

(i) MMS, EDGE, $p = 0.5$

(j) MMS, EDGE, $p = 0.8$

Fig. 5. Amazon EC2 Time performance (average, and standard deviation in errorbars).

we believe that the performance of our protocol will considerably improve under faster technologies like 4G.

Lastly, let us compare the results of our protocol with those of Orbot (Tor). First note that for less performing networking conditions (EDGE), our protocol and Orbot perform similarly, independently on α and β, while Orbot performs better for faster networking conditions (HSPA) and high values of α and β. However, high values of α and β provide better protection against anonymity breaches. In fact, Tor (and hence Orbot) does not guarantee the same level of anonymity as our approach does, since it does not provide sender protection (i.e., $\alpha = 1$) or receiver protection (i.e., $\beta = 1$) against the cellular operator. Also, if a single communication is assumed as in our setting, the communication is further exposed and $(1, 1)$-anonymity is provided by Tor, meaning that no anonymity is guaranteed to the communicating parties.

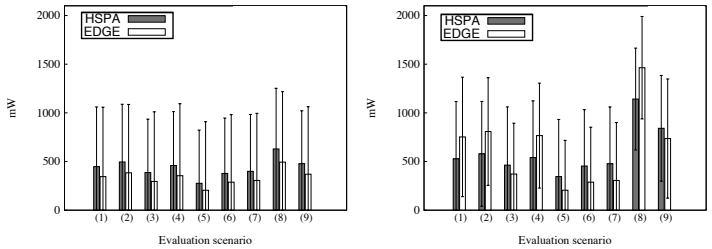
7.3 Energy Consumption

To measure the energy consumed by the smartphones using our protocol, we used the Mobile Device Power Monitor.[4] This device samples the smartphone battery with high frequency (i.e, 5000 Hz) so to yield accurate results on the battery power, current, and voltage.

We measured energy consumption for SMS and MMS sending in 9 different scenarios reported in the caption of Figure 6. Note that, due to strict space limits, we report the results related to Testbed 1 only, because: *i)* the trend of the results is similar in both testbeds and *ii)* Testbed 1 performs slightly worse than Testbed 2. The reason is that, while the

4. http://www.msoon.com/LabEquipment/PowerMonitor/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing



(a) Sending SMS messages (b) Sending MMS messages

Fig. 6. Energy consumption (average, and standard deviation in errorbars), under HSPA and EDGE network coverage, in the following scenarios: (1) $\underline{s}$ sends a message to pr (α=1), (2) $\underline{s}$ sends a message to pr (α=9), (3) $\underline{s}$ sends a message to d_z (α=9), (4) $\underline{d_t}$ sends a message to pr (α=9), (5) $\underline{r}$ gets push notification from pr (β=1), (6) $\underline{d_m}$ gets a push notification and broadcasts (β=9), (7) $\underline{r}$ receives a message from d_m (β=9), (8) $\underline{s}$ sends a message to r using TOR, (9) $\underline{r}$ receives a message using TOR. Underlined parties are the target of the energy consumption measurement.

considered devices are the same, the clones and proxy in Testbed 2 (Amazon EC2 instances) are more performing than those of Testbed 1 (local servers). We calculated battery consumption assuming 100 messages sent over a time interval of 20 minutes, under HSPA and EDGE network coverages. During the experiments, the display of the smartphones has been turned off.

The results of our experiments are reported in Figure 6. The main thing we can observe from the figure is that for the peer communicating nodes, Orbot performances (scenarios (8) and (9)) are worse compared to ours for two reasons: *i)* the number of cryptographic operations and the amount of encrypted messages exchanged between the user device and the first Tor node (entry node) are greater than the ones required by our solution; *ii)* to provide anonymous end-to-end communications, Mobile Tor users have to run two different applications on the device (Orbot and Gibberbot). Furthermore, from Figure 6(b) we can observe that in general the energy consumption is higher when MMS are sent by a device under EDGE network, in all cases where the device uses the cellular operator network. This is due to the fact that the EDGE network is slower than the HSPA one, and therefore the device must keep the cellular connection open for a longer amount of time, with a faster drain of the battery. Figure 6(a) instead shows that the overhead introduced by HSPA communications when sending an SMS is higher given the small size of the messages. In fact, EDGE has smaller overhead than HSPA when the device is in idle state.

Finally, it is worth noting that the energy consumed by senders/receivers in our protocol is about 600 Joules (500mW for 20 mins). This corresponds to about 2.7% of the battery capacity (Lithium–Ion batteries (1750 mAh, 3.7 V), 23.31 KJoule of energy when fully charged) of the smartphones of our testbeds. This energy is much less than, e.g., the energy that the browser consumes on our smartphones (more than 30% of the battery capacity according to [37]), or the overhead introduced by Paranoid Android (25% extra energy spent [7]).

7.4 Networking Costs

The energetic overhead is only one aspect of the costs of the anonymity protocol in this work. Another important aspect to take into consideration is the network traffic among the clones in the cloud—the higher the (α, β)-anonymity requirements, the larger the number of messages generated on the cloud-side during the sender and receiver communication phases of the protocol. Since a typical feature of cloud computing is "pay as you go", we studied the upper bound of the monetary costs our approach induces (i.e., for α=β=36), using the Amazon's cost calculator tool.[5]

First, we note that higher costs are required on clones c_i and c_j, which send and receive 21 MB for each MMS message. According to Amazon's calculator, a user can send (receive) up to 488 messages for free (an amount of traffic till 10GB). More heavy users will have to pay $\$0.6$ for 975 MMS messages and $\$4.2$ for 2,438 messages. Clones in the anonymity set of c_i and c_j, c_s, and c_r handle about 600 KB per MMS. So, they can forward up to 17,476 free MMS messages. Furthermore, we note that every MMS induces 1.14 MB of traffic to the proxy (and in turn to the provider of our anonymity service), meaning support for 8,752 free MMS messages. The SMS messages instead cost much less and the free limit posed by clones c_i and c_j is very high. Every forwarded SMS message induces to them 21 KB of traffic, which translates in 496,955 messages for free.

In summary, since a user's clone will (help to) send a mix of SMSs and MMSs, and will change its role over time, the number of communications in which it can be involved for free is much higher than the worst case we retrieved for c_i and c_j in MMS forwarding. Also, comparing our approach with the costs that today's carriers make us pay per SMS (MMS) messages, 0.15 Euro and 1 Euro in Italy respectively, we believe that the costs of our approach are much more convenient.

8 Conclusions

We addressed the anonymity concerns of end-to-end communicating smartphone users, where clones of the mobile devices handle part of the communication. Our solution provides anonymous end-to-end communication between two users in the network, by leveraging on properties of social networks and ad hoc wireless networks. We evaluated the robustness of our anonymity protocol, assuming each party observing the communication as honest but curious, and possibly colluding with others to uncover the identity of communicating users. Finally, we assessed the performance of our protocol by comparing it with Tor for mobile Android devices on two real testbeds including respectively 9 and 36 Android powered devices and respective clones on Amazon EC2 cloud platform.

References

[1] M. Conti, D. Diodati, C. M. Pinotti, and B. Crispo, "Optimal solutions for pairing services on smartphones: a strategy to minimize energy consumption," in *Proc. of IEEE CPSCom '12*, 2012.

5. http://calculator.s3.amazonaws.com/calc5.html

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2014.2304717, IEEE Transactions on Services Computing

[2] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proc. of MobiSys '10*, 2010.

[3] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading." in *Proc. of IEEE INFOCOM '12*, 2012.

[4] S. Kosta, C. Perta, J. Stefa, P. Hui, and A. Mei, "Clone2Clone (C2C): Peer-to-Peer Networking of Smartphones on the Cloud," in *Proc. of USENIX HotCloud '13*, 2013.

[5] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing," in *Proc. of IEEE INFOCOM '13*, 2013.

[6] M. V. Barbera, S. Kosta, A. Mei, V. C. Perta, and J. Stefa, "CDroid: Towards a Cloud-Integrated Mobile Operating System," in *Proc. of IEEE INFOCOM '13*, 2013.

[7] G. Portokalidis, P. Homburg, K. Anagnostakis, and H.Bos, "Paranoid android: versatile protection for smartphones," in *Proc. of ACSAC '10*, 2010.

[8] M. V. Barbera, S. Kosta, A. Mei, V. C. Perta, and J. Stefa, "Mobile Offloading in the Wild: Findings and Lessons Learned Through a Real-life Experiment with a New Cloud-aware System," in *Proc. of IEEE INFOCOM '14*, 2014.

[9] M. V. Barbera, S. Kosta, J. Stefa, P. Hui, and A. Mei, "CloudShield: Efficient anti-malware smartphone patching with a P2P network on the cloud," in *Proc. of IEEE P2P '12*, 2012.

[10] S. Kosta, V. C. Perta, J. Stefa, P. Hui, and A. Mei, "CloneDoc: Exploiting the Cloud to Leverage Secure Group Collaboration Mechanisms for Smartphones," in *Proc. of IEEE INFOCOM '13*, 2013.

[11] C. Ardagna, S. Jajodia, P. Samarati, and A. Stavrou, "Privacy preservation over untrusted mobile networks," in *Privacy in Location Based Applications: Research Issues and Emerging Trends*, C. Bettini, S. Jajodia, P. Samarati, and S. Wang, Eds. Springer, 2009, pp. 84–105.

[12] A. Rahmati, C. Shepard, A. Nicoara, L. Zhong, and P. Singh, "Mobile tcp usage characteristics and the feasibility of network migration without infrastructure support," in *Proc. of ACM MobiCom '10*, 2010.

[13] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second–Generation Onion Router," in *Proc. of the 13th USENIX Security Symposium*, 2004.

[14] C. Ardagna, M. Conti, M. Leone, and J. Stefa, "Preserving smartphone users' anonymity in cloudy days," in *Proc. of MobiPST '13*, 2013.

[15] L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, pp. 84–90, 1981.

[16] K. Puttaswamy, A. Sala, O. Egecioglu, and B. Zhao, "Rome: Performance and anonymity using route meshes," in *Proc. of IEEE INFOCOM '09*, 2009.

[17] K. Puttaswamy, A. Sala, and B. Zhao, "Starclique: guaranteeing user privacy in social networks against intersection attacks," in *Proc. of CoNEXT '09*, 2009.

[18] P. Mittal, M. Wright, and N. Borisov, "Pisces: Anonymous communication using social networks," in *arXiv:1208.6326*, 2012.

[19] A. Mohaisen and Y. Kim, "Dynamix: anonymity on dynamic social structures," in *Proc. of ASIACCS '13*, 2013.

[20] A. Mohaisen, H. Tran, A. Chandra, and Y. Kim, "Trustworthy distributed computing on social networks," in *Proc. of ACM ASIACCS '13*, 2013.

[21] S. Seys and B. Preneel, "ARM: anonymous routing protocol for mobile ad hoc networks," *Int. J. Wire. Mob. Comput.*, vol. 3, pp. 145–155, 2009.

[22] R. Song, L. Korba, and G. Yee, "Anondsr: efficient anonymous dynamic source routing for mobile ad-hoc networks," in *Proc. of ACM SASN'05*, 2005.

[23] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "MASK: Anonymous on-demand routing in mobile ad hoc networks," *IEEE TWC*, vol. 21, pp. 2376–2385, 2006.

[24] A. Davoli, A. Mei, and J. Stefa, "Fan: friendship based anonymous routing in wireless social mobile networks of malicious communities," in *Proc. of ExtremeCom '11*, 2011.

[25] R. Jansen and R. Beverly, "Toward Anonymity in Delay Tolerant Networks: Threshold Pivot Scheme," in *Proc. of MILCOM '10*, 2010.

[26] X. Lu, P. Hui, D. Towsley, J. Pu, and Z. Xiong, "Anti-localization anonymous routing for delay tolerant network," *Computer Networks*, vol. 54, no. 11, pp. 1899 – 1910, 2010.

[27] C. Ardagna, S. Jajodia, P. Samarati, and A. Stavrou, "Providing users' anonymity in mobile hybrid networks," *ACM TOIT*, vol. 12, pp. 1–33, 2013.

[28] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proc. of EuroSys '11*, 2011.

[29] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE TKDE*, vol. 13, no. 6, pp. 1010–1027, November–December 2001.

[30] L. Kleinrock and F. Tobagi, "Packet switching in radio channels, part ii – the hidden terminal problem in carrier sense multiple access and the busy tone solution," *IEEE TC*, vol. COM-23, pp. 1417–1433, 1975.

[31] M. Anisetti, C. Ardagna, V. Bellandi, E. Damiani, and S. Reale, "Map-based location and tracking in multipath outdoor mobile networks," *IEEE TWC*, vol. 10, no. 3, pp. 814–824, 2011.

[32] B. Levine, M. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix systems (extended abstract)," in *Proc. of FC '04*, 2004.

[33] M. Wright, M. Adler, B. N. Levine, and C. Shields, "The predecessor attack: An analysis of a threat to anonymous communications systems," *ACM TISSEC*, vol. 7, no. 4, pp. 489–522, 2004.

[34] A. Mei and J. Stefa, "Give2Get: Forwarding in Social Mobile Wireless Networks of Selfish Individuals," *IEEE TDSC*, vol. 9, no. 4, pp. 569–582, 2012.

[35] L. Buttyán and J.-P. Hubaux, "Enforcing service availability in mobile ad-hoc wans," in *Proc. ACM MobiHoc '00*, 2000.

[36] M. Jakobsson, J.-P. Hubaux, and L. Buttyán, "A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks," in *Proc. of FC '03*, 2003.

[37] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. Singh, "Who killed my battery?: analyzing mobile browser energy consumption," in *Proc. of WWW '12*, 2012.

Claudio A. Ardagna is an Assistant Professor at the Dipartimento di informatica, Università degli Studi di Milano, Italy. His research interests are in the area of cloud and information security, mobile networks, and security certification. He is the recipient of the ERCIM STM WG 2009 Award for the Best PhD Thesis on Security and Trust Management. The URL for his web page is http://www.di.unimi.it/ardagna

Mauro Conti is an Assistant Professor at the University of Padua, Italy. He obtained his PhD from Sapienza University of Rome, Italy, in 2009. After his PhD, he was a Post-Doc Researcher at Vrije Universiteit Amsterdam, The Netherlands. He has been Visiting Researcher at GMU (2008), UCLA (2010), UCI (2012 and 2013), and TU Darmstadt (2013). He has been awarded with a Marie Curie Fellowship (2012) by the European Commission, and with a Fellowship by the German DAAD (2013). His main research interest is in the area of security and privacy.

Mario Leone obtained is MSc in Computer Science from University of Padua, Italy, in 2013.

Julinda Stefa is a Post-Doc at the CS department of Sapienza University of Rome, Italy. She obtained the PhD and Laurea degree in Computer Science (summa cum laude) from the same university in respectively 2010 and 2006. Her research interests include distributed systems, system and network security, mobile cloud computing, and social mobile networks. She received a 2011 research grant from Working Capital PNI and offered by Telecom Italia (30 winners out of 2138), and the IEEE Infocom 2013 and IEEE SECON 2013 Best Demo Award.