# Security Rationale for a Cooperative Backup Service for Mobile Devices[⋆]

Ludovic Courtès, Marc-Olivier Killijian, David Powell

LAAS-CNRS, Université de Toulouse, France

**Abstract.** Mobile devices (e.g., laptops, PDAs, cell phones) are increasingly relied on but are used in contexts that put them at risk of physical damage, loss or theft. This paper discusses security considerations that arise in the design of a cooperative backup service for mobile devices. Participating devices leverage encounters with other devices to temporarily replicate critical data. Anyone is free to participate in the cooperative service, without requiring any prior trust relationship with other participants. In this paper, we identify security threats relevant in this context as well as possible solutions and discuss how they map to low-level security requirements related to identity and trust establishment. We propose self-organized, policy-neutral mechanisms that allow the secure designation and identification of participating devices. We show that they can serve as a building block for a wide range of cooperation policies that address most of the security threats we are concerned with. We conclude on future directions.

## 1 Introduction

Mobile devices (e.g., laptops, PDAs, cell phones) are increasingly relied on but are used in contexts that put them at risk of physical damage, loss or theft. However, fault-tolerance mechanisms available for these devices often suffer from shortcomings. For instance, replicating data to a storage device (e.g., USB stick or disk drive) carried along with the mobile device is risky: that device could easily be lost or stolen, or it could be damaged precisely when the mobile device itself is damaged. Data "synchronization" mechanisms, which allow one to replicate a mobile device's data on a desktop machine, are an improvement but they usually require that the desktop machine be either physically accessible or reachable *via* the Internet. Use of third-party backup servers typically also requires access to some network infrastructure.
Unfortunately, in many scenarios where devices are carried along in different places, access to a network infrastructure (e.g., *via* a Wi-Fi access point) is at best *intermittent*. Often, access to a network infrastructure may be too costly and/or inefficient energy-wise and performance-wise to be considered viable

[⋆] This work was partially supported by the MoSAIC project (ACI S&I, French national program for Security and Informatics; see *http://www.laas.fr/mosaic/*), the Hidenets project (EU-IST-FP6-26979), and the ReSIST network (EU-IST-FP6-26764).

"just" for backup. In emergency situations and upon disaster recovery, for instance, infrastructure may well be unavailable for an unspecified amount of time. In such cases, data produced on a mobile device while the network is unreachable cannot be replicated using the aforementioned synchronization techniques and could be lost. Similarly, environments with scarce Internet connectivity, such as those targeted by the "One Laptop per Child" project (OLPC, *http://laptop.org/*), can hardly rely on access to an infrastructure for doing data backup.

We aim to address these issues by providing a *cooperative* backup service, called MoSAIC [12,27]. The idea borrows from peer-to-peer cooperative services. The goal of this service is to improve data dependability for mobile devices. It leverages excess storage resources through spontaneous resource sharing among neighboring devices, using short-range wireless communications.

Anyone is free to participate in the service and, therefore, the majority of participants have no prior trust relationship. However, there are also scenarios where owners of a few cooperating devices are personal acquaintances with full trust relationships as far as the backup service is concerned (e.g., colleagues, friends, etc.). In general, an open cooperative service must be able both to account for lack of prior trust relationships among participants and to take advantage of prior trust relationships among device owners when they exist. In addition, services designed for mobile devices and *ad hoc* networks need to meet requirements related to resource constraints (energy, CPU power, network bandwidth) and intermittent or complete lack of access to a fixed network infrastructure. These constraints impose several requirements on the storage layer of our cooperative backup service [12].

In this paper, we focus on security aspects of the cooperative backup service related to secure cooperation and secure interactions between peers. We discuss their integration with the security techniques implemented at the storage layer. We propose *self-organized* security mechanisms that may be used to support behavior accountability and a wide range of cooperation policies. We show how cooperation policies can take advantage of these mechanisms to address some of our security concerns. Our approach differs from earlier work in that it focuses on *policy-neutral* security primitives that do not restrict the user's choice of a policy, rather than focusing on a specific policy.

Section 2 provides an overview of our cooperative backup service. Section 3 presents the security concerns we want to address. Section 4 provides an overview of the storage layer of our cooperative backup service. Section 5 proposes core security mechanisms and shows (i) how they fulfill some of our requirements and (ii) how they can be used as a building block for various cooperation policies. Section 6 deals with implementation considerations. Section 7 summarizes related work. Finally, Section 8 concludes and depicts on-going and future research work.

## 2 MoSAIC Overview

Our cooperative backup service, which we call MoSAIC, can leverage (i) excess storage resources available on mobile devices and (ii) short-range, high-bandwidth, and relatively energy-efficient wireless communications (Bluetooth, ZigBee, or Wi-Fi). More importantly, we expect our cooperative backup service to improve long-term availability of data produced by mobile devices. The idea is borrowed from peer-to-peer cooperative services: participating devices offer storage resources and doing so allows them to benefit from the resources provided by other devices in order to replicate their data [27]. Participating devices discover other devices in their vicinity using a suitable service discovery mechanism such as [39] and communicate through single-hop connections, thereby limiting interactions to small physical regions.

Anyone is free to participate in the service and, therefore, participants have no prior trust relationship. In the sequel, we use the term *contributor* when referring to a node acting as a storage provider; we use the term *data owner* when referring to a "client" device, i.e., one that uses storage provided by the contributors to replicate its data. All participating devices may play *both* the owner and the contributor role.

When out of reach of Internet access and network infrastructure, devices meet and spontaneously form *ad hoc* networks which they can use to back-up data. However, it would be unrealistic to rely on chance encounters between devices for recovery. Instead, we require contributing devices to eventually send data stored on behalf of other devices to an agreed-upon Internet-based store accessible by the data owners [12,27]. Once this has been done, the duty of contributing devices has been fulfilled and they can remove the data from their local store. Eventually, data owners may restore their data by querying the Internet-based store. In practice, the implementation of this Internet repository is an orthogonal issue: it could be implemented in a number of different ways ranging from a simple centralized server to a peer-to-peer distributed store.

This way of handling intermittent infrastructure connectivity makes our approach comparable to delay-tolerant networks (DTNs): data blocks that are transmitted by data owners to contributors can be viewed as packets sent to the Internet-based store and where contributors act as relays [42].

MoSAIC's approach to cooperative backup also bears some similarity with earlier work on cooperative data storage [3,26] and caching for mobile devices [22,41]. However, it differs from them in several ways. First, unlike typical distributed file system access patterns, data that is backed up is produced by a single device and may usually not be accessed by other devices. Second, unlike most caching strategies, our approach does not seek to improve locality of data replicas: instead we expect replicas to propagate to the Internet-based store, much like packets in a DTN.

Previous work studied the design of a storage layer for our cooperative backup service and compared the CPU/storage tradeoff of various data encoding schemes [12]. This study led to the storage-layer design outlined in Section 4. We also analytically evaluated the dependability of data carried on a mobile devices participating in the cooperative backup service using generalized stochastic Petri nets (GSPNs) and Markov chains [11]. This paper focuses on primitives enabling cooperation among distrustful participating devices.

## 3 Security Context and Motivations

This section details security issues that arise in a cooperative backup service among distrustful devices and concludes on security goals.

### 3.1 Threats to Confidentiality and Privacy

There is an obvious threat to confidentiality when it comes to storing critical data on untrusted devices: A malicious storage contributor may try to access data stored on behalf of other devices. Therefore, confidentiality has to be provided at the storage layer and is achieved through regular encryption techniques, as will be discussed in Section 4. Thus, communication eavesdropping is not a serious additional threat to confidentiality. Since data blocks exchanged between two participating devices are encrypted, an eavesdropper cannot gain any more information about the contents of the data being backed up than the contributor itself. Likewise, data blocks must be named by the data owner in a way that is meaningless to contributors [12] so, again, disclosing such names to a potential eavesdropper does not present an additional threat. Since the storage layer provides end-to-end encryption, the communication layer does not need to provide any additional encryption. This is a fortunate consequence since it allows CPU and energy savings to be made.
However, privacy of the participating users can be threatened. An eavesdropper may be able to know *whether* a device is actively replicating data, and it may be able to estimate the amount of data being replicated. It may also be able to know the parties involved (the physical devices or even their owner), especially when in their physical vicinity. Recent attempts to provide anonymity in MANETs, for instance based on anonymous multi-hop routing [38], appear to be relatively bandwidth-consuming and energy-inefficient. Thus, we do not address threats to privacy in this paper. However, we hope to provide a minimum level of identity privacy by allowing users to use self-managed identifying material (which may not establish any binding with their real-world identity, i.e., *pseudonyms*), rather than compelling the use of identifying material provided by a central authority.

### 3.2 Threats to Integrity and Authenticity

There are also evident threats to data integrity and authenticity: A malicious contributor could tamper with data stored on behalf of other nodes, or it could inject garbage data that would pass all the integrity checks performed by data owners but would not be of any use to the data owner.
Integrity threats also arise at the communication layer: an intruder may try to tamper with messages exchanged between two devices (essentially storage requests), thereby damaging the data being backed up. Thus, the communication

layer must also guarantee the integrity of messages exchanged between participating devices.

### 3.3 Threats to Availability

Unavailability threats against the cooperative backup service fall into two categories: unavailability resulting from accidental data loss (including accidental failure of contributors holding replicas), and data or service unavailability resulting from *denial of service* (DoS) attacks committed by malicious nodes.
Obviously, data unavailability due to accidental failures of either the owner or contributor devices is the primary concern when building a cooperative backup service.
Malicious participating devices may also try to harm individual users or the service as a whole, denying use of the service by other devices. A straightforward DoS attack is *data retention*: a contributor either refuses to send data items back to their owner when requested or simply claims to store them without actually doing so. DoS attacks targeting the system as a whole include *flooding* (i.e., purposefully exhausting storage resources) and *selfishness* (i.e., using the service while refusing to contribute). These are well-known attacks in Internet-based peer-to-peer backup and file sharing systems [2,13,29] and are also partly addressed in the framework of *ad hoc* routing in mobile networks [5,33]. These threats can be seen as *threats to cooperation.*

### 3.4 Discussion

Security threats related to the data being backed up, in particular threats to data availability, confidentiality, and integrity are largely addressed by the storage layer of our cooperative backup service. Section 4 provides an overview of the storage layer and how it addresses these issues.
Service availability is also at risk in the presence of intruders and non-cooperative participants. The very possibility of allowing malicious devices to participate in the cooperative service threatens cooperation among participants as a whole. We believe that cooperation can only be leveraged if the cooperative service supports *accountability.* In our view, accountability is a building block upon which users can implement their own higher-level *cooperation policies* defining the set of rules that dictate how they will cooperate. Section 5 proposes core mechanism as a means to provide accountability and discusses cooperation policies that may be implemented on top of it.

## 4 Architectural Overview of the Storage Layer

The storage layer presented in [12] addresses the efficient storage and indexing of data owners' critical data. It follows a write-once read-many (WORM) or append-only storage model similar to that found in archival storage systems [37], where new versions of files are appended rather than substituted to previously-stored versions. It produces a number of *data blocks*, each of which is bound to a *name* which is used to store/retrieve it to/from contributors. Since participating nodes are mutually suspicious, the storage layer provides guarantees for data confidentiality, integrity and authenticity: it supports data and meta-data encryption as well as integrity and authenticity checks, using an appropriate encoding. The general framework can be summarized as follows:

1. The data owner (rather: the cooperative backup software on the owner-side) chops the data items to be backed up into small blocks and assigns them a *block name*. A block name can be, for instance, a cryptographic hash of the block content, thereby providing *content-addressable storage*[1] [37]. An important requirement is that (i) the naming scheme must be meaningless to contributors and (ii) blocks must be encrypted. In other words, contributors cannot make any assumptions on the block naming scheme used by data owners.
2. The data owner produces meta-data blocks describing, among other things, how data blocks are to be re-assembled to produce the original data. Those meta-data blocks are themselves named in a similar way. Authenticity is achieved by signing just part of the meta-data. For instance, if meta-data blocks are the intermediate nodes of a Merkle tree whose leaves are data blocks [32], then only the root block needs to be signed, which reduces reliance on CPU-intensive cryptography; verifying the root block's signature actually allows the authenticity of the whole tree to be checked.
3. When a contributing device is encountered, the data owner sends it some of its data and meta-data blocks using remote procedure calls (RPCs). This is realized through the invocation `put (name, content)` which sends data `content` to the contributor and asks it to bind it to `name`. Since owners can choose any block naming scheme, contributors must arrange to provide per-owner block name spaces in order to avoid collisions among blocks belonging to different owners. Obviously, in order to increase data availability, data owners may choose to replicate each block [11].

The end result of this backup process is an opaque identifier that names an (encrypted) root meta-data block. We refer to this identifier as the *root block name*.
The root block name is critical since it allows all the user's data to be recovered, so it also needs to be backed up. However, as new versions of the data items (e.g., a single file or a whole file system hierarchy) are backed up, new data and meta-data blocks are created, each having a new name, and thus a new root block name is produced (this issue is not uncommon in the context of peer-to-peer file sharing and archival systems [2,37]). Consequently, data owners should store their latest root block name on contributors *under a fixed block name* to

[1] Use of content-addressable storage allows identical data blocks to be identified. Therefore, it permits the implementation of *incremental backup*, where only new blocks are transferred to contributors.

allow restoration to be bootstrapped conveniently. Since it is a critical piece of information, data owners may choose to encrypt it.

When a contributor gains Internet access (rather, when it gets sufficiently cheap or high-bandwidth Internet access), it transfers data blocks stored on behalf of other devices to an Internet-based storage server that data owners can eventually access to restore their data. That Internet store could be implemented in many different ways, ranging from a peer-to-peer distributed store to something as simple as an FTP server. However, it should support the `put` mechanism or a slightly enhanced version thereof so that both name-block bindings and per-owner block name spaces are preserved.

Restoration of backed up data typically occurs when the data owner device has failed or been lost. In this case, data owners first retrieve the root meta-data block (from the Internet-based store), decrypt it and decode it (which can only be done by its legitimate data owner), and then recursively fetch the blocks it refers to. Fetching blocks upon restoration is achieved through a `get (name)` RPC that returns the contents of the block designated by `name`.

Of paramount importance is the inability for arbitrary users to tamper with a data owner's name space on the Internet-based store. For instance, it must be impossible for a malicious user to overwrite a data owner's block associated with a specific name on the Internet repository without this being detected. However, since block encoding is owner-specific, the Internet-based store cannot check the authenticity of incoming data blocks without knowing the exact encoding scheme used by their owner. This can be solved by having the Internet-based store keep a list of all incoming data blocks associated with a given name, should different blocks be `put` under the same name (collisions). Upon recovery, the data owner can then detect and eliminate invalid data blocks in cases of collisions; invalid data blocks may be readily detected by the data owner using the possibilities offered by its encoding scheme, such as digital signature or hash verification.

It is worth noting that among the mechanisms presented here, only the actual storage protocol (i.e., the `put` RPCs) is enforced. This leaves users with the ability to choose any *security policy* for their data: they may choose any data availability, confidentiality and integrity mechanism while still conforming to the storage protocol.

## 5 Leveraging Cooperation

In this section, we present our approach to the design of mechanisms that address the threats to cooperation identified in Section 3. Core mechanisms are proposed to support accountability while being neutral with respect to cooperation policies. We then discuss issues that arise from the self-organized nature of our approach as well as cooperation policies.

### 5.1 Design Approach

There are essentially two ways to provide security measures against the DoS threats listed earlier in MANETs and loosely connected peer-to-peer backup systems: *via* a *single-authority domain*, where a single authority provides certificates or other security material to participants and/or dictates them a particular policy or mechanism, or through *self-organization*, where no single authority is relied on, at any point in time [9].

In our opinion, reliance on a common authority responsible for applying external *sanctions* to misbehaving participants as in BAR-B [1] falls into the first category. For example, BAR-B contributors *must* provide a proof that they do not have sufficient space when rejecting a storage request; similarly, upon auditing, participants *must* show the list of all blocks stored on their behalf elsewhere and all blocks they store on behalf of other nodes. Failing to do so constitutes a "proof of misbehavior" that may lead to sanctions. This raises fundamental security issues: why would one disclose all this information to some untrusted entity? Does it still qualify as cooperation among *multiple* administrative domains when a single set of rules is enforced through external sanctions? While this approach achieves strong service provision guarantees, it does so at the cost of being authoritarian and seems unsuitable for the kind of open cooperation network we envision.

Likewise, the use of so-called "tamper-resistant security modules" as in [6] can be considered a single-authority domain approach: security modules act as a local representative of an "authority" and *enforce* part of the protocol (in [6], the *nuglet* mechanism) in order to provide protection against malicious users. This leaves the user with no choice but to abide by the rules set forth by the security module and the party that issued it.

In this paper, we only focus on self-organized approaches. First, they are a good match for mobile *ad hoc* networks which *are* self-organized. Second, since we are designing an *open* cooperative service where anyone can participate, self-organization is likely to make the service more readily accessible to everyone; conversely, requiring every user to register with some central authority would be an undesirable burden likely to limit user adoption. Finally, we advocate that reliance on a central authority can in itself be considered as a security threat, to some extent: that authority is in effect a *single point of trust* and its compromise would bring the whole service down. Furthermore, depending on their security policy, users may not be willing to fully trust such an authority just because they have been told it's a "trusted" authority. They may also want to have full control over the actions that can be taken by *their* device, rather than handing over some authority over the device to some possibly unknown third party. Therefore, we prefer to focus on self-organized solutions and do not consider solutions based on a single-authority domain.

As a consequence, we cannot assume that any single cooperation policy is going to be used by *all* devices: each device can, and will, implement its own policy. We believe that the ability to choose a security and cooperation policy is particularly important when using our cooperative backup service for two reasons. First, the goal of this service is to improve the availability of users' critical data. As such, users are likely to be willing to pay attention to the contributors they deal with, and hence, they may be concerned with their cooperation policy.

Second, mobile devices being resource-constrained, users are likely to require tight control over their resource usage, and may want to implement a cooperation policy that makes the best use of their resources. This is quite different from, for instance, Internet-based file sharing services where participating devices are typically desktop machines and where, as a result, it is safe to assume that most users will be satisfied with the same default cooperation policy.
Therefore, in this paper we focus on core mechanisms allowing for accountability rather than on actual cooperation policies.

## 5.2 Providing Secure and Self-Managed Device Designation

Devices must be able to *name* each other (i) to achieve accountability and (ii) to allow contributors to implement per-owner block name spaces, as discussed in Section 4.
To these ends, device names must satisfy the following requirements. First, since we want to build a self-organized service, where no central authority has to be consulted, it must be possible for every device to create its own name or designator. Second, for the naming scheme to be reliable, device names must be *unique* and *context-free* (i.e., their interpretation should be the same in any context). Third, since device names serve as the basis of critical operations, it must be possible to *authenticate* a name-device binding (i.e., assess the legitimacy or "ownership" of a name). Authentication is needed to preclude unauthorized use of a name, as in *spoofing* attacks. Unauthorized uses of device names would effectively hinder the implementation of per-owner block name spaces and accounting mechanisms.
These requirements rule out a number of widespread designation mechanisms. IP addresses, for instance, would obviously be unsuitable to name devices since they have none of these properties (they are not context-free, especially IPv4 link-local addresses, not unique, except for IPv6 addresses, and cannot be authenticated). The designers of Mobile IPv6 (MIPv6) had similar requirements and had made the same observations. This led them to devise "statistically unique and cryptographically verifiable" (SUCV) addresses [36].
The building block for the naming scheme we are interested in (and that of MIPv6 SUCV addresses) is asymmetric cryptography. Public keys have all the desired properties as designators: they are (statistically) unique and context-free, and they provide secure naming (i.e., the name-device binding can be authenticated, thereby precluding spoofing). In practice, public keys can be too large to be used directly as designators, which is why several protocols use cryptographic hashes or fingerprints of the public keys as designators [7,36]. In order to achieve accountability, both contributors and data owners may wish to *identify* the device they are talking to, that is, to authenticate the binding between alleged name of the peer device and the device itself. In other words, *mutual authentication* is required.
It is worth noting that the entities we want to name are instances of the cooperative backup software running on participating devices and *not* people owning

the devices, nor even physical devices. Thus, the principals involved in the cooperative backup service are *logical entities* that exist and interact solely through electronic interactions among them. Therefore, authenticating the binding between one of these entities and its name (public key) boils down to verifying that that entity holds the private key corresponding to its name [19]. Doing so is simple and does not require the use of any certification authority whatsoever.
As far as the data restoration bootstrap is concerned, a practical consequence of using public key pairs to identify devices is that a user's key pair is all that is needed to bootstrap restoration, assuming its public key is also used to encrypt the root block name. That means that users must store their key pairs reliably, outside of the cooperative backup service, by copying them on a storage device under their control (USB stick, computer, or even a simple piece of paper stored in a safe place). Obviously, the device where the user's key pair is stored must not be carried along with the mobile device itself, since it could easily be lost, stolen, or damaged along with the mobile device, making it impossible to recover the data. Elliptic curve cryptography (ECC) would be handy for that purpose: it yields keys much smaller than, e.g., "security-equivalent" RSA keys; thus an ECC key pair can be as simple as a pass phrase that may be readily memorized by the user.

## 5.3 Ensuring Communications Integrity

Once a participating device has authenticated the binding between a peer device and a name, a malicious device may try to send messages and pretend they originate from another device, thereby using resources on behalf of another device. To address this issue, the integrity and authenticity of messages (i.e., RPC invocations) devices exchange must be guaranteed by the communication layer. In particular, once devices have mutually authenticated, using their key pairs, the communication protocol must guarantee that messages received at either end of the communication channel still come from the previously authenticated device. Many well-known cryptographic protocols address this issue, with different security properties.
We believe that non-repudiation is not required in our decentralized, self-managed, cooperative backup system. Non-repudiation could be used, for instance, to make sure that a device cannot deny that it sent a series of storage requests to a certain contributor. That contributor could then *prove* to a third party that it did receive those requests. However, such proofs would likely not be sufficient to be used, for instance, as part of the "history records" maintained by a reputation system (described below): they would concern only individual requests and would consequently fail to provide a sufficiently high-level view of a device's past cooperation. For instance, to prove that a data owner requested 1 GiB of storage, a contributor would need to provide a third party with 1 GiB worth of *put* requests along with the corresponding signatures. Doing so would provide more information that is necessary and would be very bandwidth-consuming, making it impractical. Thus, non-repudiation of individual messages is inappropriate in our context.

Therefore, we plan to use regular message authentication codes (such as HMACs) to provide support for message authenticity checks. HMACs can only be verified by the receiver, and therefore do not provide non-repudiation.

## 5.4 Thwarting Sybil Attacks

Since key pairs are to be generated in a self-organized way, our system is subject to the Sybil attack [16,30]: devices can change names (i.e., public keys) any time they want, which allows them to escape accountability for their past actions, including misbehavior. This attack defeats the implementation of a proper resource accounting mechanism, and consequently that of resource usage policies. For instance, a data owner can completely circumvent a per-device quota implemented by a contributor by just switching to a new key pair.

The verifiable designation mechanism proposed above cannot by itself prevent Sybil attacks. Instead it is up to cooperation policies to make Sybil attacks less attractive by providing incentives for users to keep using the same name (i.e., the same key pair). In a system where names are managed in a self-organized way, no cooperation policy can *prevent* Sybil attacks: They can only make them less effective, but evidence shows that well-designed policies can make them pretty much worthless [4,30,33].

Naturally, most reasonable cooperation policies have a common denominator: they tend to be reluctant to provide resources to strangers while being more helpful to devices that have already cooperated. However, in order to bootstrap cooperation, many policies may grant at least a small amount of resources to strangers [23]. This means that there is usually (i) a medium- to long-term advantage in keeping the same name and (ii) a short-term advantage in cooperating under a new name. Section 5.5 will show how actual cooperation policies can achieve this.

Fortunately, the impact of Sybil attacks is largely a matter of scale. With Internet-based peer-to-peer cooperative services, any peer can reach thousands of peers in a glimpse. Thus, even if it can only benefit from a small amount of resources from each peer, it may be able to quickly gain a large amount of resources. Conversely, in a cooperative service relying on physical encounters among mobile devices, it may take a long time and a great deal of traveling around before one is able to gain access to a useful amount of resources, which effectively makes selfishness less viable economically. Likewise, the impact of a flooding attack is necessarily limited to physical regions and/or groups of devices.

## 5.5 Allowing for a Wide Range of Cooperation Policies

User cooperation policies define the set of rules that determine how their device will cooperate. They are usually concerned with the stimulation of cooperation and the establishment of trust with other devices. To that end, cooperation policies can build on the accountability provided by the mechanisms presented above. We can imagine two major classes of cooperation policies: those based on the underlying social network, and those based on past behavioral observations, either private observation or shared reputation [4,23,28,33]. It is our goal to allow users to choose among these cooperation policies.

Cooperation policies based on the relationships already existing in the underlying social network can be as simple as "white lists", where the user only grants resources to devices belonging to personal acquaintances. There can also be more sophisticated policies: a user could also accept storage requests from "friends of friends", and it could accept to dedicate a small amount of resources to strangers as well. It can be argued that such policies do not scale since (i) the number of personal acquaintances of an individual is limited, and (ii) when travelling a lot, these acquaintances may be out of reach. On the other hand, social studies have provided evidence of a "small-world phenomenon" in human relationships [8,34] and algorithms have been proposed to discover chains of acquaintances among arbitrary users [9]. These studies can make cooperation policies based on a social network more relevant. Such policies, were they to insist on being able to verify bindings of keys to real-world identities, would trade privacy for improved resilience to Sybil attacks. However, similar policies may be used with pseudonyms instead of real-world identities.

Cooperation policies based on observations of past device behavior provide an interesting alternative: devices maintain "history records" of each other and make cooperation decisions using them as an input. History records can either be local to a device or they can be shared among devices—the latter is usually referred to as a *reputation* system [4,28,33]. In a reputation system, devices exchange history records and may use them as an additional hint to their cooperation decisions. Simulations have shown that shared history records are usually more efficient than private history records, especially in large networks or in the presence of a high device turnover [4,28]. However, many works that evaluate the outcome of such reputation mechanisms assume that all participating nodes use the *same* cooperation policy [4,33] (e.g., the same node rating algorithm, the same decision-making algorithm, etc.). There is no reason for this to be true. The result of using a reputation mechanism in a world where different policies are in use is, to our knowledge, an open issue. Nevertheless, reputation mechanisms do make Sybil attacks less attractive since few resources can be gained by a stranger. Devising a protocol that would allow trust information to be exchanged among principals potentially using different cooperation strategies is an open issue.

From a privacy viewpoint, maintaining such history records may be a concern when identities are bound to real-world entities, since it would allow one to know where a given person was at a given point in time. However, for users' privacy to be seriously threatened, attackers would need to *physically* track them, which the cooperative backup service could hardly be held accountable for. This is a lesser concern when identities are not bound to real-world entities.

## 6 Implementation Considerations

This section discusses implementation concerns and in particular the choice of actual protocols to achieve the goals outlined earlier.

### 6.1 Protocol Choice

While Mobile IPv6 [36] provides some of the features we need, we considered it impractical since its mechanisms are implemented at the network layer, and implementations are not widely available at this time.
Our implementation of the block store (essentially the `put` and `get` requests mentioned earlier) is based on Sun/ONC RPC [40]. ONC RPC defines the so-called "DES authentication mechanism", designed for authentication over a wide-area network; however, the mechanism does not address all our concerns (for example, its naming scheme for peers does not fulfill all the requirements of Section 5.2, and in particular does not allow name-device bindings to be reliably authenticated). The authentication mechanisms for ONC RPC defined in RFC 2695 [10] have similar shortcomings with respect to our goals. The RPCSec bindings for the Generic Security Services Application Programming Interface (GSS-API) [17] were not considered appropriate either (one reason is that most available GSS-API implementations only support Kerberos-based mechanisms, which assumes the availability of such an infrastructure).
Consequently, we decided to use the well-known Transport Layer Security (TLS), a protocol currently widely deployed on the Internet [15]. Although it was not designed with mobile computing and constrained devices in mind, we believe its flexibility makes it a suitable choice. In particular, TLS offers a wide range of *cipher suites*, which allows us to choose cipher suites that meet our resource saving constraints, such as cipher suites with no payload data encryption, as discussed in Section 3.1. TLS provides message authentication guarantees using HMACs, where, again, the HMAC algorithm to be used is negotiated between peers. TLS provides payload compression but this may be disabled (also subject to negotiation between peers). Again, disabling it allows us to save energy, especially since the data that is to be exchanged among peers is already compressed.
As far as mutual authentication is concerned, TLS provides it through *certificate-based authentication mechanisms*. While the main document [15] refers primarily to X.509 certificates, a proposal has been made to extend TLS to support authentication using OpenPGP certificates [31]. This extension is very relevant in our context for a number of reasons. First, OpenPGP certificates can be readily generated using widely available tools (e.g., GnuPG) and they are already familiar to many computer users. Second, OpenPGP certificates are already used in the context of secure electronic communications among individuals. Therefore, the use of OpenPGP certificates also allows users to easily implement cooperation policies based on the underlying social network, as outlined in Section 5.5.
OpenPGP certificates contain a lot more than just a public key. In particular, since they are primarily used to certify a binding between a public key and a real-

world person name, they contain information such as the real-world name and email address of the person the public key (allegedly) belongs to (the “user ID packets”), and a list of third-party signatures (certifications) indicating the level of trust put by other people in this name-key binding [7]. This information is only useful when implementing cooperation policies based on the social network.

### 6.2 Prototype Implementation

We have been working on a prototype implementation of our cooperative backup protocol that uses ONC RPC on top of TLS. Since ONC RPC implementations do not natively support the use of TLS as the underlying protocol, we did our own implementation. This proved to be easy to do, using raw TCP RPC client/server code as a starting point. We use GnuTLS [25] as the underlying TLS implementation since it is the only major implementation supporting the OpenPGP extension [31] as of this writing. GnuTLS is very flexible and has allowed us to actually make various specific trade-offs, such as disabling compression, choosing an encryption-less cipher suite, etc.
Initial measurements show that TLS induces little communication overhead. Handshake itself demands 2 KiB per connection in both directions (when using certificates with no signature packets), most of which stems from the OpenPGP certificate exchange. TLS’ record layer incurs little overhead (e.g., less than 30 octets per message with SHA-1-based HMACs), provided messages are at most 16 KiB large—otherwise messages are fragmented, which incurs additional overhead [15]. Although further measurements are needed, these results seem reasonable in our context.

### 6.3 On-Going and Future Work

We are currently in the process of evaluating the overhead, in terms of network bandwidth and CPU cost, induced by the use of TLS. We have also started implementing a set of cooperation policies, ranging from simplistic policies such as “white lists”, to more sophisticated policies that make use of local information of past interactions with other devices. The next step will be the implementation of a reputation system where participating devices can exchange and make use of cooperation certificates.
All these cooperation policies will need to be evaluated and compared, notably in terms of the overall level of cooperation yielded, and in terms of the resilience of the cooperative service to the aforementioned DoS attacks. Different reference scenarios will need to be identified to that end. It is still unclear which method we will choose to achieve this goal. Simulation looks appealing but may be hard to set up to faithfully reflect our system model. On the other hand, we may also try to build on the analytical evaluation of replication strategies that we conducted earlier [11]. Specifically, this evaluation uses a model of interactions among participating devices using Petri nets and Markov chains that could be extended to reflect various cooperation strategies.

## 7 Related Work

A lot of work has gone into thwarting availability threats due to DoS attacks similar to those described in Section 3.3. Most of this work was done in the area of peer-to-peer storage and cooperative backup. While our cooperative backup scheme with intermittent connectivity to the infrastructure is similar to delay-tolerant networks [42], the security of such networks is still largely an open issue [21,24]. This is partly due to the fact that most applications of DTNs, such as space mission networks, are not expected to be open for anyone to participate, which reduces the incentive to address these issues.

Fall *et al.* did propose security mechanisms permitting DTN routers to detect and eliminate disallowed traffic, and thereby avoid DoS attacks such as flooding against the DTN [20]. However, the proposed solution relies on centralized identity management and authorization: all participants are issued a key pair by an authority, along with a "postage stamp" signed by that authority indicating the allowed "class of service" for that user. Such an approach only addresses specific DoS attacks. Forms of non-cooperation such as refusal to forward a message are not tackled. We also believe that such an approach does not scale and suffers from shortcomings inherent to single-authority domain approaches, as discussed in Section 5.

In general, "trust begets cooperation". In the case of our cooperative backup service, data owners need to trust contributors to provide them the service, while contributors need to trust data owners not to abuse the service (e.g., by flooding it or by being selfish). While both issues have to do with trust establishment between owners and contributors, the literature tends to refer to both aspects using different names, such as *cooperation incentives* and *trust establishment*.

To evaluate the cooperativeness of a peer, one needs to be able to observe both its service usage and its service provision. When the cooperative service is packet forwarding or routing in MANETs, device cooperation can be evaluated almost instantaneously [4,23,33]. However, in cooperative backup services, service usage and service provision call for different evaluation techniques. First, service usage can be balanced using simple strategies such as symmetric trades [29] (i.e., pair-wise "tit-for-tat" exchanges), or "storage claims" that may be exchanged among peers [14]. Both approaches assume high connectivity among peers and are therefore unsuitable to MANETs. Second, *periodic auditing* has been proposed to establish trust in contributor service provision [1,13,14,29], but it requires peers to be reachable so that they can be challenged, which is unsuitable to the MANET context. In our cooperative backup service for MANETs, service provision can only realistically be evaluated when gaining Internet access or upon restoration.

Once service provision and usage can be evaluated, self-organized solutions usually make use of "history records" of peer behavior as an aid to cooperation decisions, as mentioned in Section 5.5. Simulations have been made to evaluate the impact on cooperation of such mechanisms when used by all participants, in the context of both private and shared history records [4,28]. In MANETs, reputation mechanisms have been proposed primarily in the context of packet forwarding for multi-hop routing protocols and route discovery [4,33].

Designation issues in a decentralized environment have been studied notably in the context of distributed programming and capability systems [35] as well as in the context of public key infrastructures (PKIs) [18,19]. The provision of guarantees for "address ownership" (i.e., having address-device bindings that can be authenticated) has also been a concern in the design of Mobile IPv6 (MIPv6) [36]. This led the authors to opt for "statistically unique and cryptographically verifiable (SUCV) identifiers". This is similar to one of the mechanisms we propose in this paper, except that we operate at the application level rather than at the network layer, which provides us with more flexibility.
Douceur *et al.* described the Sybil attack as a problem that is inherent to distributed systems using self-managed designators [16]. In [30] the authors showed that a reputation system can efficiently leverage cooperation even when self-managed designators are used.

## 8 Conclusion

We introduced a cooperative backup service for mobile devices that builds on the peer-to-peer, self-organizing paradigm largely used on the Internet. We identified security threats on such a service and listed subsequent security requirements. We have shown how a reduced set of well-known cryptographic primitives can be used to meet those requirements in a self-organized way. Our approach differs from earlier work in that it focuses on *policy-neutral* security mechanisms, rather than on a specific cooperation policy.
In particular, we advocated the use of public keys as self-managed, secure and unique designators for participating devices and discussed their use as a policy-neutral building block for a variety of cooperation policies, including a reputation system. Systems using self-managed designators are subject to the Sybil attack; therefore, we discussed the impact of this attack in our context and showed how cooperation policies can be implemented that reduce the harm that can be done. Finally, we discussed implementation issues and outlined the foundations of an implementation that uses TLS with OpenPGP certificate-based authentication.
The work presented in this paper is part of a larger design and implementation effort of a cooperative backup service for mobile devices. Our earlier work explored other aspects of the design space, particularly relating to storage tradeoffs and data encoding and compression techniques [12], as well as the evaluation of replication strategies [11]. Future work includes a detailed evaluation of some of the techniques discussed in this paper, as well as the deployment of a prototype cooperative backup service in real-world conditions.

## References

1. Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, Carl Porth. BAR Fault Tolerance for Cooperative Services. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pp. 45–58, October 2005.

2. Krista Bennett, Christian Grothoff, Tzvetan Horozov, Ioana Patrascu. Efficient Sharing of Encrypted Data. In *Proceedings of the 7th Australasian Conference on Information Security and Privacy (ACISP 2002)*, Lecture Notes in Computer Science, (2384)pp. 107–120, Springer-Verlag, 2002.
3. Malika Boulkenafed, Valérie Issarny. AdHocFS: Sharing Files in WLANs. In *Proceedings of the 2nd International Symposium on Network Computing and Applications*, April 2003.
4. Sonja Buchegger, Jean-Yves Le Boudec. The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks. In *Proceedings of WiOpt '03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, March 2003.
5. Levente Buttyán, Jean-Pierre Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. In *ACM/Kluwer Mobile Networks and Applications*, 8(5) , October 2003, pp. 579–592.
6. Levente Buttyán, Jean-Pierre Hubaux. Enforcing Service Availability in Mobile Ad-Hoc WANs. In *Proceedings of the First ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 87–96, IEEE CS Press, 2000.
7. Jon Callas, Lutz Donnerhacke, Hal Finney, Rodney Thayer. OpenPGP Message Format (RFC 2440). Internet Engineering Task Force (IETF), November 1998. *http://tools.ietf.org/html/rfc2440*.
8. Srdjan Capkun, Levente Buttyán, Jean-Pierre Hubaux. Small Worlds in Security Systems: an Analysis of the PGP Certificate Graph. In *Proceedings of the Workshop on New Security Paradigms*, pp. 28–35, ACM Press, 2002.
9. Srdjan Capkun, Levente Buttyán, Jean-Pierre Hubaux. Self-Organized Public-Key Management for Mobile Ad Hoc Networks. In *IEEE Transactions on Mobile Computing*, 2(1) , January 2003, pp. 52–64.
10. Alex Chiu. Authentication Mechanisms for ONC RPC (RFC 2695). Internet Engineering Task Force (IETF), September 1999. *http://tools.ietf.org/html/rfc2695*.
11. Ludovic Courtès, Ossama Hamouda, Mohamed Kaâniche, Marc-Olivier Killijian, David Powell. Assessment of Cooperative Backup Strategies for Mobile Devices. Technical Report 06817, LAAS-CNRS, December 2006.
12. Ludovic Courtès, Marc-Olivier Killijian, David Powell. Storage Tradeoffs in a Collaborative Backup Service for Mobile Devices. In *Proceedings of the Sixth European Dependable Computing Conference*, pp. 129–138, IEEE CS Press, October 2006.
13. Landon P. Cox, Christopher D. Murray, Brian D. Noble. Pastiche: Making Backup Cheap and Easy. In *Fifth USENIX Symposium on Operating Systems Design and Implementation*, pp. 285–298, December 2002.
14. Landon P. Cox, Brian D. Noble. Samsara: Honor Among Thieves in Peer-to-Peer Storage. In *Proceedings 19th ACM Symposium on Operating Systems Principles*, pp. 120–132, October 2003.
15. Tim Dierks, Eric Rescorla, Win Teerse. The Transport Layer Security (TLS) Protocol, Version 1.1 (RFC 4346). Internet Engineering Task Force (IETF), April 2006. *http://tools.ietf.org/html/rfc4346*.
16. John R. Douceur. The Sybil Attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS)*, pp. 251–260, Springer-Verlag, 2002.
17. Michael Eisler, Alex Chiu, Lin Ling. RPCSEC_GSS Protocol Specification (RFC 2203). Internet Engineering Task Force (IETF), September 1997. *http://tools.ietf.org/html/rfc2203*.

18. Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, Tatu Ylonen. SPKI Certificate Theory (RFC 2693). Internet Engineering Task Force (IETF), September 1999. *http://www.ietf.org/rfc/rfc2693.txt*.
19. Carl M. Ellison. Establishing Identity Without Certification Authorities. In *Proceedings of the Sixth USENIX Security Symposium*, pp. 67–76, 1996.
20. Kevin Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pp. 27–34, August 2003.
21. Stephen Farrell, Vinny Cahill. Security Considerations in Space and Delay Tolerant Networks. In *Proceedings of the 2nd IEEE International Conference on Space Mission Challenges for Information Technology*, pp. 29–38, IEEE CS Press, 2006.
22. Jason Flinn, Shafeeq Sinnamohideen, Niraj Tolia, Mahadev Satyanarayanan. Data Staging on Untrusted Surrogates. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, March 2003.
23. Christian Grothoff. An Excess-Based Economic Model for Resource Allocation in Peer-to-Peer Networks. In *Wirtschaftsinformatik*, 45(3) , June 2003, pp. 285–292.
24. Khaled A. Harras, Mike P. Wittie, Kevin C. Almeroth, Elizabeth M. Belding. ParaNets: A Parallel Network Architecture for Challenged Networks. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, February 2007.
25. Simon Josefsson, Nikos Mavrogiannopoulos. The GNU TLS Library. 2006. *http://gnutls.org/*.
26. Alexandros Karypidis, Spyros Lalis. OmniStore: A System for Ubiquitous Personal Storage Management. In *Proceedings of the Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 136–147, IEEE CS Press, March 2006.
27. Marc-Olivier Killijian, David Powell, Michel Banâtre, Paul Couderc, Yves Roudier. Collaborative Backup for Dependable Mobile Applications. In *Proceedings of 2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (Middleware 2004)*, pp. 146–149, ACM Press, October 2004.
28. Kevin Lai, Michal Feldman, John Chuang, Ion Stoica. Incentives for Cooperation in Peer-to-Peer Networks. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, 2003.
29. Mark Lillibridge, Sameh Elnikety, Andrew Birrell, Mike Burrows, Michael Isard. A Cooperative Internet Backup Scheme. In *Proceedings of the USENIX Annual Technical Conference*, pp. 29–42, June 2003.
30. Sergio Marti, Hector Garcia-Molina. Identity Crisis: Anonymity vs. Reputation in P2P Systems. In *IEEE Conference on Peer-to-Peer Computing*, pp. 134–141, IEEE CS Press, September 2003.
31. Nikos Mavrogiannopoulos. Using OpenPGP Keys for TLS Authentication (IETF Internet Draft). Internet Engineering Task Force (IETF), July 2006. *http://www.ietf.org/internet-drafts/draft-ietf-tls-openpgp-keys-11.txt*.
32. Ralph C. Merkle. Protocols for Public Key Cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 122–134, April 1980.
33. Pietro Michiardi, Refik Molva. CORE: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks. In *Proceedings of the Sixth IFIP TC6/TC11 Joint Conference on Communications and Multimedia Security*, pp. 107–121, Kluwer Academic Publishers, September 2002.
34. Stanley Milgram. The Small World Problem. In *Psychology Today*, 2, 1967, pp. 60–67.

35. Mark Samuel Miller. Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control, PhD Thesis, Johns Hopkins University, Baltimore, MA, USA, May 2006.
36. Gabriel Montenegro, Claude Castelluccia. Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2002.
37. Sean Quinlan, Sean Dorward. Venti: A New Approach to Archival Storage. In *Proceedings of the First USENIX Conference on File and Storage Technologies*, pp. 89–101, 2002.
38. Sk. Md. Mizanur Rahman, Atsuo Inomata, Takeshi Okamoto, Masahiro Mambo, Eiji Okamoto. Anonymous Secure Communication in Wireless Mobile Ad-hoc Networks. In *Proceedings of the First International Conference on Ubiquitous Convergence Technology*, pp. 131–140, Springer-Verlag, December 2006.
39. Françoise Sailhan, Valérie Issarny. Scalable Service Discovery for MANET. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communication*, March 2005.
40. Raj Srinivasan. RPC: Remote Procedure Call Protocol Specification, Version 2 (RFC 1831). Internet Engineering Task Force (IETF), August 1995. *http://tools.ietf.org/html/rfc1831*.
41. Liangzhong Yin, Guohong Cao. Supporting Cooperative Caching in Ad Hoc Networks. In *IEEE Transactions on Mobile Computing*, 5(1) , January 2006, pp. 77–89.
42. Zhensheng Zhang. Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges. In *IEEE Communications Surveys & Tutorials*, 8, January 2006, pp. 24–37.