Divalia: A Practical Framework for Anonymous Peer-to-Peer File Exchange in Wireless Ad-hoc Networks

Ryan Vogt
Dept. of Computer Science
University of Calgary
Calgary, Alberta
Canada T2N 1N4
vogt@cpsc.ucalgary.ca

Ioanis Nikolaidis
Computing Science Dept.
University of Alberta
Edmonton, Alberta
Canada T6G 2E8
yannis@cs.ualberta.ca

Pawel Gburzynski
Computing Science Dept.
University of Alberta
Edmonton, Alberta
Canada T6G 2E8
pawel@cs.ualberta.ca

Abstract

We describe Divalia*, a new peer–to–peer (P2P) file exchange framework that targets mobile wireless ad–hoc networks. The framework supports anonymous transfers, to protect user privacy, as well as pre–authentication of data to be received, to reduce the likelihood of spoofs and the resulting bandwidth wastage. Divalia is composed of two parts: the* Hash Set Dual-Request Protocol*, to challenge a would-be provider to prove the possession of the requested file; and, a form of* Direct Anonymous Broadcast*, for anonymous acquisition of information to be used for the challenge. One goal of this paper is to demonstrate how an original and secure P2P system can be constructed within the constraints and opportunities of mobile wireless ad–hoc networks through the use of well known and extensively studied component cryptographic methods.*

1. Introduction

We present a scheme for anonymous data exchanges geared to the requirements and opportunities inherent in mobile wireless ad–hoc networks. The scheme implements data transfers between peers within direct communication range, forming the peer–to–peer (P2P) component of the proposed protocol; but, it discovers information about data files using a form of controlled flooding, exploiting the ad–hoc nature of the network topology. We assume that there exists no preassigned or centralized trusted authority. Moreover, the user population is dynamic and unknown *a priori*.

We opt for P2P data transfers due to the well understood scalability problems of ad–hoc networks, as introduced by Gupta and Kumar [1] and extended by many others. Traffic routing over many hops is the key cause of these problems. Thus, we relax the notion of routing by exploiting the mobility of nodes to act as carriers of data and perform actual data exchanges during (short–lived) encounter episodes with the destination(s). This approach is in line with the observations of Grossglauser and Tse [2] for scalable ad–hoc networks that sacrifice delay for the benefit of throughput. Certainly, not all applications fit in this routing paradigm, which has come to be known as *delay tolerant networking* [3]. However, the application we consider naturally fits in the delay tolerant networking context.

Specifically, the application we consider is the delivery of large data files (often containing audiovisual data) to mobile users. The scarcity of wireless bandwidth suggests that we should avoid multiple hop routing as much as possible, while the constant mobility of nodes renders ineffective any notion of data retrieval based on single session communication with an originator/server node. The application can be thought of as follows: Mobile users indicate a set of files as items of interest. While roaming about, the mobile device can collect the desired file (plus any relevant metadata) via opportunistic short–lived partial file transfers. The entire file is not necessarily received from the same node, but rather assembled from the partial transfers over a possibly large number of encountered peers. Also, in true P2P fashion, nodes act both as clients and servers.

The delay incurred for receiving an entire file is not as important to the application as the integrity of the received data. However, we cannot assume that a user knows much more than a vaguely descriptive (humanly readable) name for the item of interest. The first challenge is to obtain more meaningful *fingerprint* information for the desired file. To accomplish this, we separate the fingerprint acquisition from the actual file data transfers. Locating fingerprint information is accomplished through the use of TTL-scoped flooding. At first, flooding may appear contrary to our scalability objective, but its inclusion is predicated on two observations: (a) by requesting information about a file

from many nodes, rather than just from single hop neighbours, tricking a node into accepting an irrelevant fingerprint as a correct one (by returning frivolous replies) requires collusion from a large number of nodes; and (b) pragmatically speaking, the fingerprint acquisition messages are short compared to the actual data transfers – hence, in absolute numbers, it is an acceptable overhead. We also hasten to add that control over the TTL can fine tune how much network overhead the fingerprint acquisition introduces.

The security objectives of the scheme are to preserve user anonymity and to avoid receiving garbage. In order to maintain a requester's privacy, a peer that does not possess the requested file is unable to send garbage, to discover what the requesting node is asking for, or even to determine if and when the particular segment requested is being transmitted. It is worth noting that by communicating with its direct peers, a requesting node's actions are inherently kept more private. Typically, albeit not always, users within that radius can make visual contact, which, though not guaranteed, can be reassuring (it plays the role of "soft authentication" in traditionally understood personal connections).

We note that existing Internet–based P2P file exchange systems operate under incomparably less stringent bandwidth constraints than in the mobile wireless environment, because they assume the existence of a wired communication infrastructure. Thus, in contrast to the environment we consider, in a wired environment anonymity can be accomplished at the expense of increased bandwidth overhead. The case in point is *mix networks* [4, 5, 6], that cryptographically guarantee the untraceability of the message source and disguise the message from all intermediate nodes. In addition to bandwidth overhead, all solutions in this class assume relative stability of the network infrastructure and pre-established trust of at least some participants, which are all absent in a mobile wireless ad–hoc network. Anonymity schemes used in certain other P2P systems like Napster and Kazaa are weak, as anonymity is achieved by obfuscation and volume rather than cryptography.

The issue of data integrity also plagues P2P systems, as there is limited confidence in the authenticity and quality of the exchanged files. Frequently, it is not possible to determine whether a downloaded file contains the desired content until after it has been entirely received and viewed. Clearly, in a wireless environment, downloading content which is not what it purported to be adds insult to injury, resulting in valuable bandwidth and battery power being wasted. To address the data integrity issue, fingerprint information of a file is collected before the actual transfers. While fingerprint information bears no stronger a guarantee that it is correct than the actual file, the relatively small size of the fingerprint information and the use of a flooding protocol allows a host to collect *multiple* fingerprints for the same file without a large overhead. The fingerprint collection can be performed over a range of time and space (the latter due to mobility) to the node's satisfaction. Consulting multiple sources provides resistance to collusion–based attacks. It is the task of the node to decide whether agreement of fingerprints (or a majority of them) from multiple sources is sufficient to warrant trusting them as being correct. Alternatively, the node can eliminate certain fingerprints by using some form of ranking (e.g., reputation–based schemes such as CORE [7]). In this paper we are interested in providing the protocol *framework* to accomplish the fingerprint acquisition, while the particulars of ranking and/or reputation schemes are left outside the scope of the paper.

The rest of the paper is organized as follows: in section 2 we introduce some general assumptions about the application, the node identities, and the mechanism of *Direct Anonymous Broadcast*. We also present an overview of Divalia. We then discuss the two distinct stages of Divalia in greater detail: fingerprint acquisition in section 3, and segment acquisition in section 4. Finally, section 5 provides some concluding remarks and directions for future research.

2. Overview and Assumptions

Suppose that a user, Alice, wishes to retrieve a file from a peer within her transmission range in an ad-hoc wireless P2P network. The requested file may reside at a subset of Alice's peers. Note that we use the term *peer* to denote all direct (single–hop) neighbours. Because peers are mobile and can be continually entering and leaving Alice's range, it is unreasonable to expect that the entire file can be downloaded from one and the same peer. It is therefore assumed that files are partitioned into *segments*, allowing the user to request and download individual segments from various peers and reassemble the segments at the end. We will use the following notation: F will denote an arbitrary file of length L bytes (indexed from 0 to $L-1$). We define the following:

- $H(F)$ denotes the hash of the entire F's contents.
- $H_{[i,j]}(F)$ is the hash of the fragment of F starting at the i–th byte and ending at the j–th byte.
- $H_i(F)$ is the hash of F's i–th segment. If the (fixed) segment size is S, then $H_i(F) = H_{[(i-1)S,iS-1]}(F)$.

We shall omit the parameter F when the file context is clear.

2.1. Addresses and Identities

In all communication between peers, we view the peer identities (coming from upper layers) as separate from data link layer addresses. By not relating the data link addresses to node identity, an extra level of security is achieved (albeit

via obscurity). Namely, the data link address used by a device can be selected at random upon powering up. As long as the data link layer addresses are not trivially short, the probability of collision with addresses of other peers within a node's range is acceptably small (and even then it can be resolved). For example, the 802.11 standard provides for Locally Administered Addresses that could, potentially, be assigned from a uniform random pool.

One can reasonably postulate that the requests and responses described in our scheme are conveyed as data link layer broadcast frames, thus obfuscating the identity of the receiving endpoint. The benefits of this approach are limited because an eavesdropping node that knows the protocol can analyze the relationship between requests and responses and thus determine the endpoints. Moreover, if all transmissions are data link layer broadcasts, all nodes will be forced to formally receive and process all receivable frames up to layer 3, which will cause unnecessary energy consumption.

Added concealment is provided through a semi-permanent obfuscated sender's address assigned randomly upon power up and remaining valid until the node shuts down. Therefore, other than when the data link layer destination has to be a broadcast address, using a locally significant data link address in the source and destination address fields hides any information about the true ID of the communicating nodes as far as the data link layer activity is concerned. (This technique is unrelated to Cryptographically Generated Addresses, e.g., as defined for IPv6 [8]).

2.2. Direct Anonymous Broadcast

To allow Alice to acquire file fingerprints anonymously, we adapt an idea presented by Stajano [9], which states that by removing the "from" field in a broadcast packet's header, the sender can essentially remain anonymous. Even though this flavour of anonymity is far from being mathematically provable, Stajano successfully argues that tracing a transmission to its sender by measuring the signal strength, timing, and phase of the transmitted packets is at least a difficult, inaccurate, and expensive operation. At the very least, a group of nodes would have to conspire to triangulate the sender and then make a visual identification of the person holding the mobile device.

Technically, the sender address field cannot be removed from the data-link frame; the data link protocol requires that an address be present. This is one reason why we earlier (section 2.1) advocated the use of random semi–permanent data link layer addresses. By disassociating logical IDs from data link layer addresses, transmissions become essentially anonymous (technically they are *pseudonymous*, where the pseudonym is the assumed data link layer address) while still being well–formed as far as the data link layer protocol is concerned.

In principle, a node's data-link address can be changed whenever there is no need to maintain data-link connectivity, e.g., upon completion of a file transfer. Note that the discovery messages are always broadcast, and hence always received by directly connected peers. It would have been possible to use different random source address for each and every transmitted frame but this would have been an overkill because responses to such transmissions would have to be broadcast data link layer frames; inflicting an unreasonably high overhead to all nearby nodes.

2.3. Protocol Stages

What Alice starts with is a file name, or a short, fuzzy description of the file that she wants. How can she be reasonably confident that the segment to be received in response to her request will contain a portion of the desired file, rather than, say, a pornographic advertisement? Clearly, if all Alice knows about the file is a few keywords, and the (unknown) sender simply claims to have the segment, Alice should be prepared for a disappointment.

To enforce the seriousness of her peer's offer to provide segments of a file, Alice must acquire more information about the requested file than a plain-text name-like description. The information possessed by Alice, i.e., the fingerprint, should allow her to challenge the sender before commencing the download. The sender should convince Alice that he/she owns the requested file segment. Moreover, to protect Alice's privacy, she must know something about the file that would let her identify it to an owner without revealing the file name to everybody in her neighbourhood.

The underlying idea of our proposed scheme is to completely separate the acquisition of the fingerprint from its presentation to a prospective segment provider (this scheme is related to the RASH protocol [10]). If Alice wants to download a file segment, she has to obtain first the requisite fingerprint. This operation is carried out in a way that preserves Alice's anonymity. In the second stage (the actual segment acquisition), Alice presents the fingerprint in a way that renders it useless to anyone who does not possess the file. In addition to playing the role of a secret file identifier, the fingerprint is also used as a challenge to which the prospective sender has to respond – to convince Alice that the offer is serious. Alice accomplishes this goal by presenting only *part of* the fingerprint to a prospective segment-provider, and challenges him/her to fill in the missing pieces. That is, it is imperative that the entirety of the fingerprint data Alice collected does not fall into the hands of users unable to provide the corresponding file segments.

The operation of acquiring the fingerprint, by the virtue of being relatively cheap (in terms of bandwidth) and safe (in terms of privacy), can be repeated several times, possibly at different times and in different environments. It can

also be arbitrarily far separated in time from the segment acquisition stage. Thus, before deciding to download the segment from an unknown peer, Alice can establish considerable faith in the authenticity of the fingerprint. An attack on the privacy of the complete file acquisition would require a collusion of a large number of users over a long period of time. As Alice's identity is unknown during the fingerprint acquisition stage, such a collusion would be very difficult.

Each of the two stages needed for a successful segment download can be divided into two steps. In particular, the fingerprint acquisition stage consists of 1) the request for a fingerprint issued by Alice and 2) the reply to Alice's request arriving from nodes that possess the file. Similarly, the download phase starts with 3) Alice's request for a segment matching a given fingerprint and 4) the actual segment transmission. Each of these four steps needs to be separately safeguarded. We will not deal here in detail with the privacy of the fourth step as it can be readily accomplished using well known cryptographic techniques (e.g., the Station-To-Station Protocol [11]). We shall focus on the first three steps. One should note that each of the four steps can be repeated on-demand, and as often as necessary, to cope with the intermittent character of mobile wireless connections. Naturally, a peer can play both the role of sender and receiver of data files. That is, once Alice has completely received a file, she may very well offer the file to others.

3. Fingerprint Acquisition

The first stage of the scheme allows Alice to transform the name of the file she is searching for into a fingerprint. Alice starts with the file name or description, and ends up with the following: the hash value for the complete file H, a hash value for each segment of the file, $\{H_i\ \forall i\}$, and hash values of two random fragments of the file $H_{[j,k]}$ and $H_{[m,n]}$. We define a fingerprint for the file F as:

$$P(F, j, k, m, n) = \{j, k, m, n, H_{[j,k]}, H_{[m,n]}\}\ .$$

The values H and $\{H_i\ \forall i\}$ are supplementary to the fingerprint, and are used after receiving a segment to confirm that Alice actually received the desired segment or file correctly.

If anonymity were not a concern, the fingerprinting would not be needed. Consequently, the fingerprinting itself must be implemented in a way that preserves the anonymity of Alice's request. Needless to say, if Alice just broadcast the file name for which she wanted a fingerprint, she would reveal her intentions to all peers in her neighbourhood. Note that any obfuscated variant of the file name would be equally revealing of Alice's intentions. For example, the hash of the file name, or even the hash of the entire file (assuming they are known via a directory lookup or preexisting advertisements) would be equally weak. This is because anybody could easily obtain them (in the same way as Alice) and store them in dictionaries, e.g., for the purpose of fooling requesting peers into accepting trash. However, Alice cannot tell in advance which of her current peers will respond to her query. Hence, the request must be unavoidably broadcast, and is thus exposed to all nearby nodes.

3.1. Basic Fingerprint Acquisition

We start by presenting a simple protocol by which Alice can acquire a fingerprint for a file she wishes to retrieve. There will be several security flaws with this basic protocol; however, for clarity of exposition, we present this basic protocol first, then describe how the security holes can easily be closed in the subsequent section (section 3.2).

When Alice wants to acquire a fingerprint $P(F, j, k, m, n)$, she *anonymously* broadcasts to her peers the following query:

$$Q_{Alice} = \{\textit{filename}, \text{TTL}, q, a, b, c, d\},$$

where a, b, c, and d are random integers chosen by Alice, q is a random nonce, and TTL is a time-to-live initialized to some random value in $[T_{min}, T_{max}]$. When Bob receives such a request, he does one of two things. If he has no file matching the *filename*, he decrements the TTL counter, and, if it is still positive, he rebroadcasts the request to his neighbours. Otherwise, i.e., if TTL has reached zero, Bob does not forward Alice's request.

If Bob happens to possess a file F of length L bytes named N (where N matches the *filename* specified by Alice, to within some heuristic that is beyond the scope of this paper), he replies by broadcasting (anonymously):

$$R_{Bob} = \{q, \text{TTL}, N, L, H, \{H_i\ \forall i\}, j, k, m, n, H_{[j,k]}, H_{[m,n]}\}$$

where L is the length of the file and TTL is set to a random value in $[T_{max}, T_{max} + \Delta]$. Note that the TTL setting of Bob's reply is guaranteed to be at least as large as Alice's initial value, to give the message a good chance of making it all the way back to Alice (even if Alice has moved in the meantime). Finally, $G(L, a, b, c, d) = \{j, k, m, n\}$ is a deterministic, globally-known function G with the property that $j<<k$, $m<<n$, and $0 \leq j, k, m, n<L$.

By having Alice choose the four random numbers a, b, c, and d used for selecting the two portions of F to hash, she can be certain that the hashed portions of F are actually random. The role of G is to account for Alice not knowing the length of the file when she chooses her four random numbers; however, since she learns L at the end, she can confirm that Bob has honestly converted a, b, c, and d into j, k, m, and n by applying G to her own selection of numbers. We discuss choices for the function G in section 3.2.

When a node receives Bob's message, it checks whether it has seen previously, within some time interval, a request containing the nonce q. If so, the node will rebroadcast the reply; otherwise, it will ignore Bob's message.

This way, Alice is able to gain information from distant nodes about files she wants to download, without revealing her identity. Similarly, Bob can provide the information without revealing his identity. This concept of a chain of broadcasts with unknown endpoints is based on the transmission model of Freenet, a wired peer-to-peer system [12].

Two caveats to this protocol have to be addressed. First, Bob may reveal to his neighbours that the peer providing the fingerprint is in their immediate vicinity, because his neighbours do not hear him retransmit the original request. Therefore, it would be prudent for Bob to first rebroadcast Alice's request and then, after a randomized delay, broadcast his answer. Similarly, Alice should rebroadcast Bob's reply – to disguise that she happens to be its recipient.

The second issue is that Alice's fingerprint information is now public knowledge – anyone along the path from Bob to Alice will know $P(F, j, k, m, n)$, even though they may not possess the file sought by Alice. It is essential that nodes unable to provide Alice segments from a given file do not acquire all of Alice's fingerprint information for that file. A solution to this problem is discussed next.

3.2. One-Time Keys

Alice can prevent everyone along the path from Bob to her from discovering her entire fingerprint by using a one-time public-private key pair. Ahead of time – perhaps when she is docked, so energy is no concern – Alice can generate a large set of public-private key pairs. For a given pair, let O denote the open (public) key and U the secret key.

In her initial broadcast, Alice includes O, i.e., she sends:

$$Q_{Alice} = \{\textit{filename}, \text{TTL}, q, a, b, c, d, O\} \ .$$

Let l be the bit length of the binary representation of each of the four random numbers $a \ldots d$. Let $abcd$ be the bitwise concatenation of those numbers and let M be a MAC function (a keyed hash function) with an output of length $4l$ bits. Bob computes $\epsilon\zeta\eta\theta = M_O(abcd)$ and responds with:

$$R_{Bob} = \{q, \text{TTL}, N, L, H, \{H_i \ \forall i\}, j, k, m, n, \\ E_O(H_{[j,k]}, H_{[m,n]})\} \ ,$$

where:

$$G(L, \epsilon, \zeta, \eta, \theta) = \{j, k, m, n\}.$$

What does this scheme buy us over the basic scheme of section 3.1? Let us consider two attacks that Mallory could attempt to launch against this protocol:

First, let us consider an eavesdropping attack. Suppose that Mallory is listening along the path from Bob to Alice. Mallory will hear the values of a, b, c, d, and O, as well as j, k, m, and n. We do not consider useful to encrypt j, k, m, and n because Mallory may know (or at least come up with reasonable guesses for) L due to the distribution characteristics of file sizes in general [13]. Mallory could compute candidate j, k, m, and n values on her own. However, Mallory is not in possession of U, so she cannot decrypt $H_{[j,k]}$ and $H_{[m,n]}$. At best, Mallory may be able to use the result of Bob's mapping of $\{a, b, c, d, O\}$ into $\{j, k, m, n\}$ to discover the value of L, provided G can be inverted as such. However, Mallory could just send out her own fingerprint request to learn L. Hence, Mallory learns nothing useful from an eavesdropping attack.

Next, let us consider an active attack, in which Mallory attempts to trick Bob into revealing $H_{[j,k]}$ and $H_{[m,n]}$ to her. Obviously, if Mallory were to playback Alice's request – substituting O', one of her own public keys, for O – she would receive a different reply from Bob, since the output of the function M (and hence the function G) would be different. However, we must consider the possibility that Mallory could produce values a', b', c', d', and O' such that:

$$G(L, \epsilon', \zeta', \eta', \theta') = \{j, k, m, n\}$$

for:

$$M_{O'}(a'b'c'd') = \epsilon'\zeta'\eta'\theta' \ .$$

However, doing so would require Mallory to break the MAC function M – that is, she would have to produce five input values a', b', c', d', and O' that map to a chosen output under M. Assuming that we chose a cryptographically-secure MAC function, this attack is not a concern.

It is important to note that the security of this scheme depends on the cryptographic reliability of M, not of G. In fact, G can be as simple a function as we wish, so long as $G(L, \epsilon, \zeta, \eta, \theta) = \{j, k, m, n\}$ with $j{<<}k$, $m{<<}n$, and $0 \leq j, k, m, n{<}L$. One such function that is easy to calculate and sufficient for our purposes is shown in Figure 1, where exchange(x,y) is the trivial swap operation. It should be clear that the function produces indices in the range from 0 to $L - 1$ and that $j{<}k$ and $m{<}n$ (by at least γ). To avoid looping for certain "incidents" of L with respect to γ, a practical variant of G should use several choices of γ (and even α and β) values selectable based on L.

4. Segment Acquisition

We can now assume that Alice has access to fingerprint information for the file F that she wishes to download. Specifically, she knows: $H_{[j,k]}$ and $H_{[m,n]}$, where j, k, m, n are random and $j{<<}k$ and $m{<<}n$. Also, she has H_i, the

$j \leftarrow \epsilon$;
$k \leftarrow \zeta$;
$m \leftarrow \eta$;
$n \leftarrow \theta$;
repeat
  $j \leftarrow \alpha \cdot j + \beta \mod L$;
  $k \leftarrow \alpha \cdot k + \beta \mod L$;
  $m \leftarrow \alpha \cdot m + \beta \mod L$;
  $n \leftarrow \alpha \cdot n + \beta \mod L$;
  if $j>k$ **then**
    exchange(j,k);
  end if
  if $m>n$ **then**
    exchange(m,n);
  end if
until $j + \gamma<k$ and $m + \gamma<n$

Figure 1. An example $G(L, \epsilon, \zeta, \eta, \theta)$ function.

hash of the i–th segment of F, $\forall i$, as well as H, the hash of the contents of F.

Suppose that Alice needs the i–th segment of F. It might seem that the proper way for her to proceed is to request the i–th segment of the file having hash value H. The benefit of using this scheme is that Bob[1] will send Alice a segment from his copy of F only if the contents of the file are the same as the one for which Alice is looking. In fact, Bob will send the segment even if his copy of the file has a different filename than the one known to Alice.

However, this scheme only works if Bob is an honest participant. What happens if Alice is communicating with mean-spirited Mallory? First, using a dictionary of hash values to files, Mallory can determine which file Alice is seeking. Also, nothing prevents Mallory from telling Alice that she has a file with hash value H when really she does not. Thus, Mallory could send garbage to Alice, and Alice would have to rely on confirming the value of H_i after receipt to determine if Mallory actually sent her the file segment she requested. If Mallory did not, Alice would know not to request files from Mallory any longer; however, the bandwidth and battery power would be irreversibly wasted.

4.1. Basic Segment Acquisition

The hash request scheme can be improved by exploiting Alice's knowledge of $H_{[j,k]}$ and $H_{[m,n]}$. When Alice wishes to download the i–th segment of F from Bob, she challenges Bob by presenting a portion of $P(F, j, k, m, n)$ to him. Namely, she sends the following challenge:

$$C_{Alice} = \{j, k, m, n, H_{[j,k]}\} .$$

If Bob has a file matching C_{Alice}, he responds with the value $\mu(H_{[m,n]})$, where the function μ is as described in the upcoming section 4.2. If Bob provides the correct value, Alice proceeds, i.e., she informs Bob which segment she wants, then downloads the i–th segment of F from Bob.

What does this scheme buy us? Let us again assume that Alice is communicating with Mallory instead of Bob. If F is sufficiently large, Alice's first request regarding $H_{[j,k]}$ will not reveal to Mallory which file Alice is seeking, unless Mallory actually possesses the file. This is because, for a sufficiently large F, it is impractical to build a dictionary from j, k, and $H_{[j,k]}$ to F. In fact, Mallory will not even be able to proceed with the protocol unless she is in possession of F, since she will be unable to produce $\mu(H_{[m,n]})$.

This protocol can be supplemented with additional measures to protect the privacy of Alice and Bob. One such measure is the *Station-To-Station Protocol* [11]. The Station-To-Station Protocol is similar to Diffie-Hellman, except that each peer must have both a public-private key pair and a certificate signed by a trusted authority binding the node's identity to its public key. It should be noted that using the Station-To-Station Protocol provides better protection against Man-In-The-Middle Attacks than the Interlock Protocol [14], because the Interlock Protocol is only useful when one node can tell if its peer is being impersonated by a man-in-the-middle; it is not applicable in a wireless ad-hoc scenario where one would be constantly communicating with new, unknown nodes.

Using the Station-To-Station Protocol, Alice and Bob can agree on a shared secret key for a symmetric encryption algorithm, which they may subsequently use to encrypt their transmission. The requisite public-private key pairs could be set up, hardwired, and certified by the manufacturer, or created by the user and certified by temporarily plugging the device to a solid infrastructure network, i.e., the Internet. While this type of cryptography would not protect Alice if she initiated communication with Mallory (or vice versa), it would offer two honest participants protection against eavesdroppers.

4.2. Defining the Function μ

Recall from section 4.1 that when Alice challenges Bob by presenting C_{Alice}, Bob must correctly respond with $\mu(H_{[m,n]})$, for some as-yet undefined function μ.

First, note why it is important that a function μ be applied to Bob's reply. Were Bob to simply reply with $H_{[m,n]}$, there would be an insecurity in the system. Imagine that Bob does not possess F. In theory, he should be unable to reply

[1] Note that while "Bob" is the standard name of Alice's peer, the Bob in this scenario is (most likely) different from the Bob(s) participating in the fingerprint acquisition stage.

to Alice's challenge. However, were Bob to replay Alice's challenge to Carol, who has F, Bob would learn the proper response to the challenge from Carol. He could then replay the response to Alice, convincing her that he possesses a file that he actually does not.

To prevent this class of replay attacks, Alice and Bob must each generate a random bit stream of fixed length. Specifically, Alice generates a random bit stream, X_{Alice}, and Bob generates another bit stream, X_{Bob}. After the two parties commit to their random bit streams (Schneier discusses several commitment schemes [11]) and exchange them over the encrypted channel, they define:

$$X = X_{Alice} \oplus X_{Bob} \ .$$

Using this X value, the function μ can now be defined for any string of bytes z:

$$\mu(z) = M_X(z) \ ,$$

where M is a MAC function. This definition of μ ensures that the proper reply to a given challenge is unique each time the challenge is made, thereby thwarting the aforementioned class of replay attacks.

4.3. Caveats of the Basic Protocol

It is important to realize that this protocol is still not perfect. If Mallory is in possession of F, she will still learn what file Alice was requesting. Furthermore, nothing prevents Mallory from sending garbage to Alice instead of the i–th segment of F, provided that she can produce $\mu(H_{[m,n]})$. However, the protocol will at least reduce the number of attackers who can send garbage to Alice, and Alice still has the value of H_i to confirm that she was not sent garbage.

Faced with this issue, we may ask whether it is possible for Alice to request a file in such a way that Mallory would not know what file is being requested, *even* if she has the file. Of course, the answer to this question is no. Even were such a scheme possible, Alice would *eventually* have to reveal to her peer, be it Bob or Mallory, in which file she is interested (otherwise, her peer would be unable to transmit the file to her). Hence, any two-party P2P transfer protocol suffers from this deficiency of the basic scheme: it is eventually revealed to Mallory what file Alice wants, provided Mallory has the file, and she could send some small amount of garbage to Alice before being detected.

The other problem to note is that Bob must be willing to spend cycles to compute $H_{[j,k]}$ for each file in his library for every request he receives. If this becomes too computationally-expensive, he may be forced to perform the computations on only a random sample drawn from his library at each request, thus possibly missing some requests that he could in principle fulfill. The next section investigates how this deficiency can be somewhat mitigated, at least.

4.4. The Hash Set Dual-Request Protocol

Recall from section 4.1 that requests for file segments (as well as the actual transfers of those segments) occur over a channel encrypted with a symmetric algorithm, using a key established with the Station-To-Station Protocol. The binding of a public-private key pair to a node by a certificate in the Station-To-Station Protocol provides us with a way to improve the basic segment-request scheme. In the new *Hash Set Dual-Request Protocol*, when Alice wants to request some segments from Bob, she first constructs the set

$$\mathbb{C}_{Alice} = \{C_1, \ldots, C_\nu\} \ ,$$

where each C_i is a challenge for a file F_i from which Alice wants some number of segments. Note that in principle, for each C_i a different set of j, k, m, n could be used because the fingerprint acquisition step described in section 3.1 can be repeated an arbitrary number of times to collect fingerprint information for the same file, or it can be extended to acquire information for a number of (j, k, m, n) tuples in one request. The Hash Set Dual–Request Protocol requires that $\nu \leq K$ for some pre-determined global constant $K>0$. That is, processing $\mathbb{C}_{Alice}$ will require Bob to search for a matching file no more than K times.

Then, Alice signs the random bit stream X that she created with Bob (described in section 4.2), and sends both the request $\mathbb{C}_{Alice}$ and the signed bit stream $D_{Alice}(X)$ to Bob. Note that the signed bit stream has no meaning to any peer in the network other than Bob – to Carol, $D_{Alice}(X)$ would just appear to be random garbage, and Bob could not convince her that it was anything else. Having responded to one request from Alice, Bob refrains from processing other requests from the same source for a certain amount of time. This restriction limits the computational impact of the request protocol on Bob. What is important about Alice's public key being certified in this context is that she cannot easily generate a new public-private key pair (effectively changing her identity), and issue another request to Bob before her time-out period is complete. That is, the use of the certificate prevents *Sleep Deprivation Torture* [9], in which Mallory repeatedly issues expensive requests to Bob to run his battery dry.

5. Conclusions

We have presented a framework for anonymous file exchange in a mobile ad-hoc wireless environment. In the spirit of delay tolerant applications, we have indicated how

the mobility of the nodes and the ad–hoc nature of the topology can be exploited to collect fingerprint information for a set of desired files and how it can then be used to assemble together the requested files from short–term opportunistic data transfers. The proposed scheme is based on well known cryptographic component algorithms and methods. The contribution is mainly in how to tune the anonymous P2P transfers to fit the context of intermittent connectivity, low bandwidth, and lacking pre–existing trust relationships. For example, compared to RASH [10], fingerprint information is collected on a per-file basis (as opposed to a per-segment basis). Though Divalia requires more computational effort from Bob during the segment acquisition stage, the benefit is reduced traffic over the anonymous channel.

In a vein similar to how throughput can be improved by exploiting mobility [2], we believe that mobility can also increase the degree of confidence in P2P data transfer integrity. The options are certainly not exhausted with Divalia's fingerprint acquisition scheme. Indeed, one could imagine other schemes, for example, where a host already possessing a file pretends that it wishes to download it, in order to determine the truthfulness of other nodes. There are numerous other extensions which allow for nodes to refine their trust relationships as the set of neighbouring nodes continuously changes due to mobility (the trust extensions described in RASH would also work well with Divalia). In this sense, Divalia has only scratched the surface.

As far as Divalia is concerned, there are still some open issues with its design. First, even though Mallory cannot determine $H_{[j,k]}$ and $H_{[m,n]}$ from listening to Alice's initial request for file information and Bob's reply, she can still modify the anonymous communication in transit without any consequences. Similarly, Mallory could transmit a meaningless reply to Alice's request. Alice's only defense is to take any information she receives over the anonymous channel with a grain of salt, and to re-request file information if she believes the information she received was incorrect (for example, if she cannot find any peers with a file matching $H_{[j,k]}$). For example, one strategy is to re-acquire the file fingerprint information at different points in time and space before forming segment acquisition requests.

Another technical issue with the system described in this paper is that if Bob has downloaded only some of the segments of a file, he may be unable to transmit those segments to other nodes who request them. That is, even though he may have segment i of file F which Alice is requesting, he may be unable to produce $H_{[m,n]}$ without being in possession of the entire file (or, at least more of it than he currently owns). An enhancement of Divalia in a direction that will circumvent this problem is currently under research. Also, once Bob is in possession of an entire file and can produce any requested $H_{[m,n]}$, an index (or other data structure) to facilitate fast lookups based on $H_{[j,k]}$ becomes necessary.

References

[1] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[2] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4):477–486, August 2002.

[3] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of ACM SIGCOMM 2003*, pages 27–34, August 2003.

[4] D. Boneh and P. Gole. Almost entirely correct mixing with applications to voting. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, pages 68–77, November 2002.

[5] M. Jakobsson. A practical mix. In *Proceedings of EUROCRYPT 1998*, pages 448–461. Springer-Verlag, (LNCS 1403), May/June 1998.

[6] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proceedings of the 18th IEEE Symposium on Security and Privacy (SP '97)*, pages 44–54. IEEE Computer Society, May 1997.

[7] P. Michiardi and R. Molva. CORE: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Advanced Communications and Multimedia Security, IFIP TC6/TC11 6th Joint Working Conference on Communications and Multimedia Security*, pages 107–121, Portoroz, Slovenia, September 2002.

[8] T. Aura. Cryptographically generated addresses (CGA). *draft-aura-cga-00*, February 2003.

[9] F. Stajano. *Security for Ubiquitous Computing*. John Wiley and Sons, February 2002.

[10] R. Vogt, I. Nikolaidis, and P. Gburzynski. No junk, no peeking, serious offers only: P2P file exchange in wireless ad-hoc networks. In *Proceedings of the 30th IEEE Conference on Local Computer Networks (LCN '05)*, pages 10–17, Sydney, Australia, November 2005.

[11] B. Schneier. *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., 1996.

[12] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Protecting free expression online with Freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[13] A. B. Downey. The structural cause of file size distributions. In *Proceedings of ACM SIGMETRICS 2001*, pages 328–329, June 2001.

[14] R. Rivest and A. Shamir. How to expose an eavesdropper. *Commun. ACM*, 27(4):393–395, April 1984.