EnPassant: Anonymous Routing for Disruption-Tolerant Networks with Applications in Assistive Environments †

Gauri Vakde Radhika Bibikar Zhengyi Le Matthew Wright

The University of Texas at Arlington

Dept. of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019, USA

Summary

Disruption-Tolerant Networking holds a great deal of potential for making communications easier and more flexible in pervasive assistive environments. However, security and privacy must be addressed to make these communications acceptable with respect to protecting patient privacy. We propose EnPassant, a system for using disruption-tolerant networking in privacy-preserving way. EnPassant uses concepts from anonymous communications, re-routing messages through groups of peer nodes to hide the relation between the sources and destinations. We describe a set of protocols that explore a practical range of trade-offs between privacy and communication costs by modifying how closely the protocol adheres to the optimal predicted path. We also describe the cryptographic tools needed to facilitate changes in group membership. Finally, we present the results of extensive trace-based simulation experiments that allow us to both compare between our proposed protocols and observe the costs of increasing the number of groups and intermediate nodes in a path. Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: Disruption-tolerant networking (DTN), anonymity, wireless, pervasive, privacy

Prepared using secauth.cls

Security Comm. Networks **00**: 2–26 (2010)

DOI: 10.1002/sec

Security Comm. Networks **00**: 2–26 (2010)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/sec.000

1. Introduction

Assistive environments using sensors, PDAs, handheld devices, and other pervasive technology require communications between devices that is flexible and robust. The primary model for this kind of communications is *disruption-tolerant networking (DTN)*, a framework in which messages are routed through the network by taking advantage of periods when two nodes are connected [1, 2]. For example, suppose that a nurse carries a PDA equipped with an 802.11 wireless receiver. The nurse passes by a set of sensors, equipped with 802.11 wireless transmitters that send their sensor readings to the PDA. The PDA then comes within the vicinity of a laptop, and both devices are Bluetooth enabled, so the PDA uploads the sensor readings to the laptop. Finally, the laptop is plugged into a wired network and uploads the sensor readings to a server for storage and processing. The communication provided by message-passing in this fashion saves the expense of putting in a complete networking infrastructure. This is particularly important to facilitate creating assistive environments in developing countries, sparsely populated rural areas, and for quick deployment in established facilities.

Assistive environments, however, have critical privacy requirements. In particular, data security and privacy must be maintained to protect patients from having their health information exposed. This makes the use of the kind of message passing used in DTN an unacceptable privacy risk – most DTN systems do not limit messages from being sent to only known and trusted parties [3]. In fact, if untrusted parties are never used to forward messages, messages may wait for long periods without being sent and may fail to be delivered. Encryption can and should be used to hide the contents of messages. However, the fact that a source is contacting a particular destination may itself be sensitive. For example, if a patient sends a message to a specific doctor, then the doctor's specialty (e.g. cancer or heart disease) can reveal a likely set of diseases that the patient may have.

In this paper, we explore a method to hide the relationships between the source and destination in opportunistic networks for assistive environments. In particular, we propose EnPassant,[†] a framework that applies the principles of anonymous communications to the specific challenges of opportunistic networking. The basic idea of EnPassant is to identify a series of intermediate receivers, called *pawns*, and send the message through this series of pawns towards the destination. The message will be encrypted in layers, following the basic idea of Chaum's mixes [4] and used in the popular Tor system [5], such that each pawn can see only the next pawn in the series. The series of pawns is effectively an overlay route and the system can use a variety of routing techniques to get the message from pawn to pawn, including those of

*Correspondence to: 416 Yates St., Arlington, TX 76019, USA
†This work was sponsored in part by National Science Foundation CAREER award CNS-0954133 and award CNS-0916221

Prepared using secauth.cls *[Version: 2008/03/18 v1.00]*

† The name comes from the chess maneuver called *en passant*, in which a pawn is captured while passing by an opposing pawn. In DTNs, packets that are passing in the network may be caught be the attacker and their privacy must be maintained.

[1, 2].

In DTNs, however, communication is already uncertain and slow. If we were to route messages through a series of randomly pre-selected nodes, too many messages may never be delivered or take unacceptably long to arrive. To provide a better tradeoff between providing privacy and keeping both failed deliveries and networking overhead low, EnPassant puts nodes into groups, with group members sharing public key pairs. This allows the source to select a series of intermediate *pawn groups* to forward the message. When a message is sent to a particular pawn group, it can be passed through the system until it reaches any member of the group. This reduces overhead by allowing the message to be routed to the nearest member of the pawn group, rather than to a specific node. In this paper, we shall discuss these tradeoffs and enhancements.

We now outline the rest of this paper. In the next section, we will provide some background on routing in DTNs and discuss related work. In Section 3, we describe a simple network model for DTNs and the possible attacker types, these basic models guide the design of EnPassant. In Section 4, we will describe the EnPassant system and ways to integrate the pawn groups with established ways to perform routing in DTNs. We outline a key-distribution framework in Section 5 that help make EnPassant easier to deploy and maintain. In Section 6 we describe our communication scenario and the overall simulation experiment. We present the results of our simulations to measure the privacy and network cost tradeoff in Section 7. In Section 8, we outline two additional schemes that make EnPassant more flexible by providing a choice in balancing privacy and performance. We conclude in Section 9 with a discussion of some possible future directions.

2. Background

Disruption-tolerant networks (DTNs) — also known as delay-tolerant networks — are an evolution from the model of mobile ad-hoc networks (MANETs). Both of these models share the idea that users, enabled with wireless devices, can communicate by using intermediate nodes — i.e. other users' devices — as routers. This allows the users to communicate in areas with little or no infrastructure. MANETs and their counterpart, vehicular ad-hoc networks (VANETs), are best suited to applications such as military operations and busy highways, in which users are densely packed and the network remains mostly well-connected.

DTNs, however, allow for the possibility that the users may move out of signal range for some time, leading to partitions of the network. By making robust data forwarding and queueing algorithms, DTNs can deliver messages even when users are spread apart and have only intermittent contact. One application for DTNs is *pocket-switched networks*, studied extensively in the HAGGLE project (`http://www.haggleproject.org`), in which users carry devices on their person and the devices

Prepared using secauth.cls
DOI: 10.1002/sec

exchange messages with Bluetooth [6]. The DakNet project [7] showed how rural villages in India could be connected using shared kiosks and mobile access points (MAPs) attached to busses, motorcycles, and bicycles that go between the village and Internet-connected towns.

2.1. Routing in Delay Tolerant Networks

To understand how we can apply anonymous routing techniques in DTNs, we now discuss DTN routing mechanisms. DTN messages can be packaged as *bundles* that are routed in a store-and-forward manner though the network [8]. We can model a DTN as a graph composed of nodes connected by links. The availability and capacity of links are time-varying. There can be multiple links between a pair of nodes. A contact in the context of DTNs is an opportunity to send data over a link. DTNs may be constrained by limited resources like contact opportunities, contact duration, data carrying capacities of contacts, storage at nodes, processing abilities of nodes, and the energy capacity of each node.

The main goal of DTN routing, unlike fully connected data networks, is to maximize the chance of delivery. However, there are no intuitive metrics that we can use directly for building routes with this goal in mind. However, minimizing delivery latency can reduce the chance that a message gets dropped. This approach was validated in simulation by Jain et al [1].

Routing algorithms in DTNs can be classified as either flooding-based or forwarding-based. Flooding-based algorithms replicate each message and send it to many nodes. The redundancy implies reliability in terms of eventual delivery of the message as it can be reasoned that at least one copy will reach the destination. These approaches also seem to reduce latency. However, these algorithms can be very expensive in terms of consumption of resources. They are also not scalable. Epidemic routing [9] is a flooding-based routing approach. In this approach, each node stores its messages in its buffer and when the two nodes come in contact with each other they exchange summary vectors that list the IDs of the messages in their buffer. Then the nodes exchange messages to synchronize their buffers. Several variants have been suggested to make epidemic routing less expensive.

On the other hand, forwarding-based algorithms use knowledge of the network topology to determine the optimal path along which the message must be forwarded. Jain et al. describe algorithms that use knowledge oracles, abstract entities representing specific knowledge about the network [1]. Four different oracles are defined — the contact summary oracle, the contacts oracle, the queuing oracle, and the traffic demand oracle. The contact summary oracle gives time-invariant, aggregate statistics about the contact schedule. For example, it can provide the average waiting time until the next contact of an edge. The contacts oracle can give any information

DOI: 10.1002/sec

regarding contacts between two nodes at any point in time. The queuing oracle gives instantaneous buffer occupancies at any node at any time; it is most difficult oracle to realize. The traffic demand oracle gives the present or future traffic demand. These oracles may not be practical, as they require globally-distributed knowledge of the network, but they provide a way for us to design and understand DTN protocols.

As the latency depends on the time a message arrives at a node, this time is taken into account to compute the cost of links emanating from that node. The paper presents four algorithms based on a time-varying version of Dijkstra's algorithms, these are: Minimum Expected Delay (MED), Earliest Delivery (ED), Earliest Delivery with Local Queues (EDLQ), and Earliest Delivery with All Queues (EDAQ). These algorithms use progressively more information from the oracles.

To make routing more practical, one can make a routing algorithm self-configuring and more suitable for imprecise or unpredictable schedules [2]. Jones, et al. propose the Minimum Estimated Expected Delay (MEED) protocol where the link-cost is based on the observed contact history over a sliding history window.

In this paper, we have suggested using information from such routing protocols to improve the performance of the EnPassant system. Specifically, our scheme requires that the routing protocols can be queried by a node to find its distance to other nodes. The distance could be based on any relevant metric like bandwidth or latency. Our scheme relies on the information provided by these protocols and thus works only as well as the information they use. However, our scheme can leverage new advances in such protocols and thus become more efficient with improvements in them.

2.2. Related Work

The first anonymous communication solution for DTN proposed in [3] uses identity-based cryptography (IBC). It suggests making identities of the users more specific by combining them with geographical identifiers. Further, it recommends hierarchical identity-based cryptography (HIBC) for greater scalability. The paper attempts to protect user identities from DTN routers (including kiosks) but it is assumed that DTN gateways are trusted and are aware of user identities. Protocols designed therefore use pseudonyms instead of user identities while routing messages in DTNs. For example, to send a message a user must compute a pseudonym through which a router can ensure that the user is valid but can't obtain the associated identity. Commonly in DTNs one-way authentication is required for which the authors have suggested a quick, non-interactive scheme. Additionally, an anonymous mutual authentication scheme which takes three flows is also given.

The advantage of this scheme is that it incurs no additional overhead for routing. However, the scheme is very tightly tied to the DakNet model, and it assumes a strongly trusted central authority.

Prepared using secauth.cls
DOI: 10.1002/sec

A more general approach is required for systems in which a trusted central authority cannot be assumed. This is especially true in assistive environments, where having a trusted authority presents a substantial privacy risk for patients.

Implementing security features in DTNs have been an ongoing topic of research. Key distribution is a significant problem; some work has proposed using IBC [10, 11]. A Bundle Security Protocol has been proposed through the IETF [12]. This protocol could be a way to implement some of the cryptographic constructions that we propose in this paper.

Many systems have been developed for anonymous communications on the Internet, including Tor [5], Mixminion [13], and Freenet [14]. Tor [5] is a widely-used system for anonymizing TCP connections, for applications such as Web browsing, in which the streams are passed through a sequence of proxy servers. Tor uses layered encryption, also called *onion routing*, from which we derive the main idea of our scheme. Mixminion [13] is a system for anonymizing email that is also based around the onion routing mechanism. In the sense that DTN messages have more in common with email than Web browsing, allowing for long delivery delays, Mixminion may be closer in some respects to our approach. However, Mixminion proxy servers are always connected and therefore are capable of sending cover traffic and other measures to stop powerful attacks. Freenet [14] is a system for anonymous peer-to-peer file sharing. Unlike most anonymity systems, including EnPassant, Freenet does not use onion routing. Instead, Freenet hides user identities by passing requests and files through other peers. Similar to our approach, Freenet does not seek to protect against powerful eavesdroppers; instead, it provides practical privacy against more limited attackers at acceptable system cost.

3. Network and Attacker Models

Before describing our proposals for protecting privacy, including personal medical information, while using disruption-tolerant networks (DTNs), we first outline the basic models we are using to guide our design. We describe a simple network model for DTNs that is detailed enough to compare different algorithms. We also describe possible attacker types and the extent to which we seek to defend against them.

3.1. A Model for Disruption-Tolerant Networks

As described in Section 2, there are a variety of applications for disruption-tolerant networking, from remote village Internet connectivity to pervasive computing environments. One example that has been used in a variety of experiments is a city bus system, with both stationary access points and moving busses with occasionally crossing routes [1]. Since there are network traces available for this model, we will also use it as the basis for our work. We expect that our results will apply well in general, but recognize that some applications may feature unique networking characteristics that will require more specific models.

DOI: 10.1002/sec

For example, we assume that all nodes know about all other nodes; this assumption does not scale to large and dynamically changing networks. However, this is also true for the underlying routing protocols we apply [1]. We note that the EnPassant framework can be extended to different routing protocols that do not make this assumption, but exploring how to make those protocols efficient is beyond the scope of this work.

In our model, we have a set of nodes, some of which are stationary and others physically move in the local area. When two nodes are in proximity, within the range of their respective wireless transmitters and receivers, the nodes may exchange a number of messages depending on their transmission rates and the amount of time they are in proximity. While the exact transmission rate may depend on the nodes' distance from each other and may vary from connection to connection, we model the transmission rate as a reasonable fixed value across the system and across time. In our experiments, we model the system more simply by allowing each sender to send all of the messages it chooses based on other considerations like queue size and desired route. This will cause us to underestimate latency and message overhead, but will still allow us to make reasonable comparisons between algorithms. Because we are only concerned with whether two nodes are in contact, we extract connectivity information from traces and ignore the actual movement patterns of the nodes.

As we describe in Section 4, we put nodes together in pawn groups. We select nodes for each group at random from all nodes in the system. We group the nodes unequally by placing each node into a group (the total number of groups is pre-chosen) chosen at random (with replacement), leading to varying group populations. This is more realistic than having fixed group size, as groups are best selected as nodes that trust each other, at least in a limited way. Note that group members will not be assumed to trust each other to protect each other's privacy or otherwise provide any specific help to group members. However, group members will share keys and it is easier to establish shared keys among trusted group members. For example, with sensors or busses administered by the same administrator, the administrator can be in charge of generating the group key. Among people with strong trust relationships, a single trusted friend could be the key generation authority, or the authority could rotate in round-robin fashion through the group. If a semi-trusted EnPassant authority assigns nodes to groups, then the group should either elect a key-generation authority or rotate the authority among the nodes in a round-robin fashion.

3.2. Attacker Model

While the privacy of users is important, we cannot protect it absolutely against attackers of unlimited capability, at least not with reasonable cost. Thus, we must carefully consider the capabilities of the attacker we seek to protect against. We will now describe

DOI: 10.1002/sec

several such attacker models that we believe to be reasonable.

3.2.1. Local Eavesdropper

One of the more likely attacker types is an attacker in the vicinity of the user with the ability to eavesdrop on the user's wireless signals. This attacker could, for example, simply be a curious neighbor. Such an attacker requires little in the way of equipment — a simple wireless receiver is sufficient to observe messages in the system. Fortunately, the attacker is easily defeated. In a DTN, most messages travel over multiple hops far away from the local eavesdroppers' wireless receiving range. The attacker can see the packet go from the user to the first hop, but is unlikely to be able to track the packet beyond that point. By simply encrypting messages and the final destination with a key that the attacker doesn't have will prevent most leaks.

3.2.2. Curious Responders

Another likely attacker is the responder who receives the messages from the initiator. This may seem counter-intuitive, as the responder and initiator exchange messages. However, the responder knows the content of the messages and may be more curious than any other party about who sent them. This attacker, like the local eavesdropper, is easily defeated. As long as the messages that reach the responder contain no information about the initiator, little can be revealed.

It is possible that, with detailed knowledge of the network layout and expected message delivery latencies, the responder could learn something from the delay between messages. For example, if a response message is expected immediately after receiving a message from the responder, the round-trip latency can be measured. Similar attacks have been explored for the Tor network [15]. However, without a realistic deployment, it is hard to perceive exactly what kind of information could be leaked. Further, the attack depends greatly on the application being used and does not work against messages whose sending start times are not highly predictable. Thus, we leave exploration of this line of attack to future work.

3.2.3. A Set of Compromised Peers

Another attacker type that should be anticipated is a set of compromised peers that work together to break the privacy of users. At the very least, a single compromised peer should be defended against, representing an otherwise honest but curious member of the network. However, a more aggressive attacker could insert a set of compromised peers into the network. For example, the administrator of a set of sensors or busses could use these nodes to monitor some of the activity in the network.

An attacker with a set of peers can perform a variety of attacks. First, with a presence in multiple groups, he will have access to multiple private keys, enabling decryption of more messages than any single node. Second, the compromised peers can record

DOI: 10.1002/sec

observations of which users send messages, the times that they are sent, and the recipients of messages during that time. If a user sends many messages to the same destination or set of destinations over the time, its statistical patterns will stand out and the attacker will be able to link sources and their destinations. Similar attacks have been described in anonymous communications for the Internet [16, 17, 18, 19]. We intend to apply similar analytical and simulation methodologies to evaluate our proposed approaches.

3.2.4. Partial Eavesdropper

Another important attack model is the *global eavesdropper*, an attacker with the ability to see, for all messages in the network, the sender, receiver, and sending time. We argue that a fully global eavesdropper is not appropriate for our study. DTNs are designed for environments in which regular connectivity is not available. Other wireless networks, such as sensor networks and connected ad-hoc networks, may be subject to global eavesdroppers due to the nodes being densely populated in a more compact area [20, 21]. In DTNs, however, nodes are expected to be without connectivity due to large distances relative to their wireless transmission range. In such a network, the eavesdropper would need a very large ratio of wireless receivers to nodes in the network to cover the entire network's operations.

Nevertheless, a *partial eavesdropper* would be able to place wireless receivers in a few *hotspots* in the network where many nodes meet and exchange messages. From observing these hotspots, the attacker could learn much about what is happening in the network. Similar attacks against Tor were explored in [22], which explores the threat of a compromised Internet exchange (IX) that can monitor a substantial fraction of the network's traffic. In such networks, the IX will observe many of the same communicating pairs over time. In DTNs, however, the eavesdropper may see different communications and have different capabilities for attack. We plan to explore the power of these attacks to de-anonymize users.

3.2.5. Combined Attacks

Finally, we recognize that an attacker need not be limited to a single vector of attack. For example, a local eavesdropper and a curious responder would be able to combine their collected information to better expose the initiator. Such attacker types must also be considered in the system design.

3.3. A Novel Metric for Measuring Privacy in DTNs

There have been a variety of ways to measure the privacy of anonymous communications systems. One well-known approach is the use of information theoretic metrics, as proposed by [23, 24]. Generally speaking, this will be the number of bits of entropy from the perspective of the attacker. Low entropy means that little information is hidden from the attacker, while high entropy means that the attacker can discern very little about the network.

DOI: 10.1002/sec

Unfortunately, given the variety of attack models possible, it is difficult to estimate the attacker's general knowledge about the network, his knowledge of the current locations and sending patterns of other nodes, and his ability to track messages. This makes an absolute measure of privacy impossible. Instead, we aim to measure the relative privacy that different EnPassant variants provide in a way that is meaningful for multiple attacker models.

To this end, we have designed a metric based on path similarity. Specifically, we recreate a network scenario several times and have the same sender send a message at the same point in time during each run of the scenario. If the system provides very high privacy, the path taken by the message will vary substantially in each run. In contrast, if the system provides little privacy, the message may take the same path or a highly similar path every time. We can explain this in terms of the attacker's ability to de-anonymize messages. If the the path is highly similar, it is also highly predictable. Thus, an attacker who observes a message later in the path will have a greater ability to retrace where it came from and thereby better narrow down the list of possible initiators. If the paths are more divergent, then the attacker has a harder time of guessing the initiator.

To convert this intuition into a metric, we label each node in the path with character and apply the Levenshtein distance [25] between pairs of strings. The Levenshtein distance is an edit distance that measure the minimum number of single-character edits (insertions, deletions, or substitutions) to go from one string to the other. Given a set of s strings, all generated from the same scenario, we calculate the Levenshtein distance between all pairs and take the average similarity percentage (distance divided by the length of the first string). A higher similarity percentage indicates more path similarity and lower privacy; a lower similarity percentage indicates greater privacy. Note that we only consider delivered messages in our metric.

4. System Design

The EnPassant system applies the principles of anonymous communications to disruption-tolerant networking, with the goal of enhancing privacy for applications like patient care. In place of mixes or onion routers, we propose pawn groups, a group of nodes of which any could serve as an intermediate pawn in the overlay path. We now describe the elements of this system: the basic cryptographic framework and the integration of routing with pawn group and pawn selection.

4.1. Group Onions

We call the layered cryptography used to provide anonymity *group onions*. Consider an initiator I sending a message M_{IR} to a responder R, using pawn groups A and B. Suppose that for a given pawn group X, the pawn group shares a public key KU_X and a private key KR_X. Any member of the group can decrypt a message M encrypted with the public key

DOI: 10.1002/sec

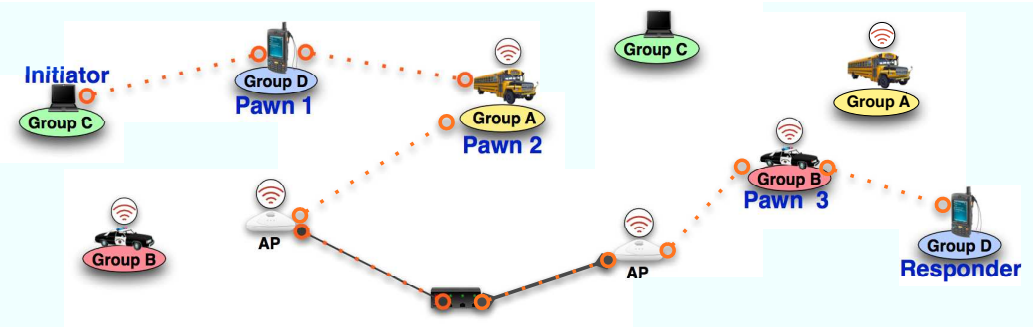

Fig. 1. EnPassant Architecture. The dotted line shows the path for a message from the initiator to the responder.

as $E_{KU_X}[M]$. Let us assume that the public keys of all pawn groups are known to all nodes in the system, including I. Also, note that nodes are members of only one pawn group each. This is to prevent a single node from decrypting more than one layer of the group onion.

To send its message, I will construct a group onion by progressively encrypting the message with the public keys of each of the pawn groups it wants on its path. Within each layer, it will put the ID of next pawn group to which the message should be sent. We now briefly show the sequence of messages.

First I sends M_A to pawn group A. Let us use the notation $X \sim\!\!\rightarrow Y$ to indicate that node X sends a message to a member of group Y. The selection of that member and the process of routing the message are described in Section 4.2:

$$I \sim\!\!\rightarrow A : M_A = E_{KU_A}[M_B, B]$$

Suppose that node A_0, a member of A, gets M_A and becomes the first pawn on the path. A_0 decrypts M_A to get M_B and the ID of group B. It then sends M_B to a member of group B.

$$A_0 \sim\!\!\rightarrow B : M_B = E_{KU_B}[M_{IR}, R]$$

Then suppose that B_0, member of B, gets M_B and becomes the second pawn on the path. B_0 similarly decrypts M_B to get M_{IR} and the ID of responder R. It then sends M_{IR} to R. From these messages, we see that M_A can be written as:

$$M_A = E_{KU_A}[E_{KU_B}[M_{IR}, R], B]$$

Through this process, the identity of the sender, I, has been hidden from any intermediaries except the first node to receive the encrypted message from I. Any node with the ability to see M_{IR} (which may or may not be encrypted depending on the message or application) or the identity of the responder R will not know the identity of I.

4.2. Routing and Node Selection

To take advantage of the fact that EnPassant uses pawn groups instead of selecting individual nodes, we need to devise a scheme to route messages to a group of

DOI: 10.1002/sec

nodes. To clarify, this is different from multicasting, in which the message is sent to all nodes in the group. Rather, the message should go to any member of the group, as with anycast in IP networks. In this section, we will describe a series of possible ways to route the message through the path of pawn groups and discuss the expected impacts of each choice on the performance and privacy of the system.

4.2.1. Random Selection

The simplest way to build the EnPassant system would be to not consider the way in which the underlying routing algorithm works, and to use it as a black box without modification. This provides a substantial advantage in keeping the system modular, allowing for different underlying routing algorithms to be used without any changes to EnPassant itself. To do this, EnPassant initiators would have to be agnostic to performance considerations when selecting pawns. Because of this choice, we would be assured that no privacy would leak to do biased selection of pawns due to performance considerations. Other research has examined ways in which privacy can be lost when performance is used to select onion routers in the Tor network [26, 15, 27]. However, we can also expect that ignoring the underlying routing algorithm will provide the worst performance, although significantly better than directly selecting individual pawns.

We now outline a simple algorithm that would operate without knowledge of the underlying routing algorithm. In this *Random Selection* scheme, the initiator first chooses each pawn group uniformly at random. Then it chooses the initial pawn uniformly at random from the first pawn group. As the message is forwarded to a member of each group, that member chooses the next pawn uniformly at random from the next pawn group. For example, if the initiator is using a path of two pawns, it will select the first pawn group G^1 and then select a second pawn group G^2 *without replacement*. In other words, we are careful not to select $G^2 = G^1$ so to ensure that the same key could not be used to open both layers of encryption and see the message. Having selected the groups, the initiator chooses a pawn, say G_i^1, from among the nodes in G^1 and sends the message to it using the standard routing algorithms. When G_i^1 gets the message, it decrypts it and sees that the next pawn group is G^2. It then selects the next pawn, say G_j^2, at random from G^2 and sends it the message. G_j^2 decrypts the message and sends it directly to the responder R.

Note that we allow each pawn to choose the next pawn in the path from among the next pawn group. This may seem undesirable if, for example, the first pawn is corrupt and has a corrupt partner in the second pawn group that it can choose. However, if there are such corrupt partners, they can also share their groups' private keys with each other, making it unnecessary for the first pawn to select the second pawn to see the responder's identity. The security of the system is thus based more on the selection of groups than individual nodes. Note that the pawns have no impact on the selection or order of pawn groups, as the initiator

DOI: 10.1002/sec

encrypts the message with the public keys of the pawn groups that it chooses.

4.2.2. Closest Pawn Routing

Although Random Selection has the benefits of being modular and not leaking any privacy, it also does not help to limit overheads. Since DTNs can be quite slow, we would seek to provide better performance as long as not too much privacy is leaked. We now describe *Closest Pawn Routing (CPR)*, a scheme that uses information directly from the routing protocol to improve performance. For this scheme, we assume two things about the underlying routing algorithm: (1) that the routing algorithm maintains some notion of *distance* between nodes based on metrics such as latency or bandwidth; and (2) that we can query the routing algorithm to find out the distance between the query node and all other nodes in the system. Several existing routing algorithms for DTNs meet these requirements, including ED and MED [1].

Given these two requirements, we can select pawns from each group more intelligently. We first assume that the initiator selects the pawn groups randomly, as with Random Selection. Then, each sender selects as the pawn the closest member of the next pawn group in the path. By doing this, we are making a greedy choice that ignores future pawn groups, but we expect to improve both message overhead and average path latency over Random Selection (which also ignores future pawn groups during pawn selection). In particular, we expect that if the distance from the current sender to the closest member of the next pawn group is half that to the average pawn group member, then the overall cost of sending the message will be cut in half. Note that the cost reflects the routing algorithm's cost metric and may not correspond exactly with the cost metric we seek to minimize. However, the cost metrics for both normal routing and EnPassant should be roughly the same for most systems.

Using CPR will results in some privacy loss compared with random selection. The main attack is to take advantage of the locality the choice of pawns provides. While random selection should not have any locality, CPR will allow an attacker to observe that the user's messages are closer to the user than to other nodes in the system. For example, let us assume that the last pawn is always physically located within radius r of the initiator (even accounting for node movement). If the user's messages can all be identified as coming from the same initiator, then the attacker could take the intersection of all nodes with radius r of the last pawn for each observed message. Eventually, such intersection attacks can substantially limit the set of possible initiators [28].

This attack is overly simplistic, but it suggests the direction that an attacker could take to gain an advantage in CPR that random selection would not allow. One defense against this would be to use random selection for the final pawn; the locality-based attack would fail unless the attacker controlled the final two pawns for multiple messages. Another

DOI: 10.1002/sec

approach is simply to ensure that the ratio of groups to nodes is high enough so that most selected pawns will not be close to the initiator.

4.2.3. Directed Routing

In CPR, we seek to improve performance by having each sender intelligently select the closest member of the next group as the next pawn. The improvement in performance is achieved by lowering the overall cost of sending the message. However, no effort is made to direct the message towards the responder which may result in selection of a path that eventually doesn't lead to the responder or that has pawns which are closer to the sender but farther away from the responder.

Therefore, to further improve routing performance, we propose Directed Routing (DR), a scheme that directs the EnPassant packets along the optimal predicted path obtained from the underlying routing algorithm for routing a message from an initiator to a responder. We define the optimal path as the one with the lowest cost, as identified in the underlying routing algorithms. We expect that directing the EnPassant path along the optimal path will not only decrease the overall cost of sending a message but it will also result in more messages being actually delivered to their final destination. To implement this scheme we require the underlying routing algorithm to support CPR and capable of being queried, at the source node, for the optimal path for routing a message. Again ED and MED [1], meet these requirements.

In DR, the initiator first queries the routing algorithm to obtain the optimal path for routing a message to the responder. Note that different routing algorithms have different ways of determining the optimal paths and thus the paths may differ for any two algorithms. From the path, the initiator identifies the intermediate nodes that eventually lead to the responder. The initiator finds out which groups these intermediate pawns belong to and selects the groups from among the list. The group of the first intermediate node becomes the first intermediate pawn group for EnPassant, the group of the second node becomes the second group for EnPassant and so on and so forth till the initiator gets the required number of intermediate pawn groups.

Note that a group will appear only once in the list of intermediate pawn groups. Also, the order of appearance of groups in the list will be in accordance with the order of the corresponding nodes in the optimal path. If the number of nodes in the path is less than the required number of pawn groups, then the extra groups are chosen randomly as with Random Selection and they are inserted in the list after the groups which are selected from the optimal path information. Again, if the routing algorithm does not find a path between the initiator and the responder, then also we choose all the intermediate pawn groups randomly. In DR, as with CPR, each sender selects as the pawn the closest member of the next pawn group in the path.

The improved performance for this scheme also

Prepared using secauth.cls
DOI: 10.1002/sec

means greater loss of privacy. Clearly, it will aid in the locality-based attack that is applicable to CPR. Moreover, it will aid a curious responder in finding the initiator by tracking the direction from which the messages arrive. In Section 8 we discuss some variations to this approach so that the EnPassant path is partially directed by the routing algorithm while providing some improvement with respect to privacy.

4.3. Design Considerations

We shall now discuss some of the design considerations for EnPassant. EnPassant creates two ways to trade off between privacy and performance; we shall briefly discuss these tradeoffs. We shall also discuss the benefits and costs of enhancing EnPassant through the use of cover traffic.

The first way to trade off between privacy and performance is the size of the group. Large groups make it easy to find a member nearby, increasing the chance of quickly passing the message to the destination. However, with few groups, there are fewer possibilities for diverse paths needed for anonymity over the lifetime of the connection. Also, there is the possibility of a single node stealing private keys from a few other nodes and thus being able to open all the layers of communication for a set of messages. With more groups, the danger of this attack is reduced.

The second way to trade off between privacy and performance is with the number of pawn groups a sender selects for its path. Clearly, shorter the path, lower the overhead and delay. Very short path lengths, however, lead to lower privacy for the sender. For example, with a path length of only one pawn group, a pawn in that group will be in position to see the intended recipient and possibly the source node as well. With enough messages, the source will eventually be linked with the messages. Further, the messages will all be received by pawn nodes close to the position of the source node, helping to reveal the sender's location. If the source selects two pawn groups in its path, this will improve privacy by enforcing greater distances from the source and clearly separating the message from the source node. In general, the more intermediate pawns on the path, the harder it is to link the source and recipient.

To enhance the privacy of EnPassant further against powerful attackers like the partial eavesdropper, the nodes can use cover traffic. When a node comes in contact with another node, it should send all available real messages first to ensure high delivery rates. If the contact is still available, it could then send dummy messages. This would cost power consumption for the sending node that generates and transmits dummy messages and for the receiving node that receives and filters dummy messages. When power consumption costs are a prime consideration, these overheads may be too high. In some cases, however, they may be acceptable. This simple scheme provides variable costs and benefits, depending on connection times and traffic rates, among other parameters. Thus, the tradeoffs require further analysis.

Prepared using secauth.cls
DOI: 10.1002/sec

5. A Key-Distribution Framework for EnPassant

In this section, we propose a key-distribution framework that we can use to make group onions possible. The special issues we need to address in DTNs are listed as follows.

- **Node connectivity.** Nodes aren't always online and can't get new keys all the time.
- **Dynamic groups.** Group membership may need to be updated as nodes leave or join the system.
- **Resilient Security.** The system security must recover even though a portion of the secret is exposed.

We observe that, for the purpose of providing privacy in DTNs, using an old or revoked key for a short time is not a major risk to users. The idea of our approach to address the above challenges is to update the keys periodically without immediately revoking older keys. Groups' public keys will be made available periodically, e.g. through periodic distribution of certificates via a gossiping protocol or simply placed on servers if most users can access the server periodically. A protocol for secure gossiping of information in mobile environments was proposed by Quercia et al. [29], and this could be used to distribute the certificates without any being left out. Then the keys will be updated, derived from the prior key, fairly frequently (e.g. once a week). Note that, depending on the security needs of the system and the capability of each node of getting the new keys, the key update period can be less frequent or we can even use static keys. According to our model for DTNs described in Section 3, the key distribution framework works as follows.

A set of stationary access points are in charge of public key distribution and they have their regular public/private key pairs, KU_{AP_k} and KR_{AP_k} for access point k. KU_i^t is denoted as the public key for group i during time period t and KR_i^t is the corresponding private key. When node i approaches an access point k at the end of a time period t, it will:

- Request the update of the public keys of all pawn groups.
- Present the request (R) for its new group private key with group member ID (ID), signed with its current group private key and encrypted with the public key of the access point.

$$i \sim\!\!\rightarrow k : M_k = E_{KU_{AP_k}}[E_{KR_i^t}[R], ID]$$

- Receive its new private key if access point k verified its signature and there were no conflicting requests.

Access point k will perform the following tasks:

- Generate group key pairs periodically and distribute them to all other access points. We assume that access points are wired so that data transmission is fast and inexpensive.
- Handle requests from nodes, verify their signatures, and ensure that there are no conflicting requests.

Prepared using secauth.cls
DOI: 10.1002/sec

- Sign and distribute the requests it received to all other access points.
- Encrypt the new private key with the old one, sign it, and then send it to the EnPassant node.

$$k \rightsquigarrow i : M_i = E_{KR_{AP_k}}[E_{KU_i^t}[KR_i^{t+1}]]$$

This key distribution framework is designed to prevent an attacker from using compromised keys to get renewed ones. If they are going to do this, the access point will receive two requests from the same group member ID so that key exposure is detected, and the key will be disabled.

6. Simulation Design

To evaluate the performance of the proposed EnPassant protocols we used DTNSim2[‡] [30], a discrete-event simulator for DTN that uses FIFO, reliable links with fixed bandwidth and delay. It is based on the DTN simulator developed for the DTN routing paper by Jain et al. [1].

Our aim is to implement the EnPassant protocols over forwarding-based routing algorithms for DTN to evaluate their relative privacy, in terms of average similarity distance, and performance, in terms of the delivery ratio, average delivery delay, and overhead in terms of actual bytes transmitted per each byte of the data. Moreover, we want to study the manner in which performance is influenced by the average size of the groups in the system, and the number of pawn groups an initiator selects for its path. For our purpose, it is sufficient to implement the protocols over ED and MED which are both forwarding-based DTN routing algorithm that require some form of schedule information as described in Section 2.

We shall now describe our communication scenario and the overall simulation.

6.1. Scenario: City Buses

Scenarios based on a network of buses equipped with wireless devices providing DTN have been previously discussed by Jain et al. [1] and Jones et al. [30]. In [30], Jones et al. note that the trends for all aspects that they had studied were similar for a city bus scenario and a wireless LAN scenario. However, for performance under ideal conditions it was noted that, though the trends were same, the city bus scenario had a better performance.

We therefore use the DieselNet traces [31] that were compiled for buses running routes serviced by UMass Transit in the Amherst, MA region. UMass Transit had a total of 40 buses with DieselNet mobile networking equipment.

The buses that would operate on a day depended on which ones were available and which ones had to be rested due to breakdowns or maintenance requirements. As a result, different buses operated on different routes on different days; hence, the schedules for any two days are not identical. Every bus connected with other buses or access points (APs) when it would come in contact with them. The traces

[‡] Available at `http://watwire.uwaterloo.ca/DTN/sim/`

Prepared using secauth.cls
DOI: 10.1002/sec

thus contain connection events between buses as well as between a bus and an AP. Connection events for a day are stored in a file matching that date. To make the simulations tractable, we eliminate all but the ten most commonly-contacted APs. Thus, we create a scenario with less infrastructure than in [1] and [30].

6.2. Traffic

In [30], it is mentioned that for a bus scenario, the schedules of five weekdays had been combined because these schedules were identical and statistics for messages generated during the second day were recorded. Since the daily schedules obtained from DieselNet trace are not identical, we can not apply this scheme to our scenario. We instead concentrate on obtaining statistics for traffic sent over a single day that has high probability of being delivered on that same day. We do this for five consecutive days and calculate the final statistics as a suitable average.

For generating traffic for a day, we listed all ordered pairs of nodes in that schedule. The first node in the pair is the initiator and the second is the responder. Pairs having both the initiator and responder as APs were discarded as such a communication would happen via the Internet rather than the DTN. For each of the remaining ordered pairs, we select a random start time from the first quarter of the day. At this time, the initiator will send a message to the responder for the first time. That initiator will continue sending a message to the same responder at intervals of one hour during the first half of that day. The size of each message is 10,000 bytes. This pattern ensures that each node initiates at least six messages to every other node that can be a valid responder in the DTN. Since this traffic is generated in the first half of the day, it has high probability of being delivered on the same day.

Note that sending messages later in the day might be delivered the next day, but our simulation cannot extend to the next day to see whether this occurs or not. In general, our traffic pattern is meant to provide enough traffic in the network to see how the protocols operate.

6.3. Metrics

We evaluate our protocols in terms of path similarity (discussed above in Section 3.3), delivery ratio, average delivery delay, and the overhead. Delivery ratio is the percentage of total messages delivered to their final destinations by the end of the simulation out of the total messages injected into the system. Delay for a delivered message is the time between when that message is generated at the initiator and when it is received at it's final destination. By the end of the simulation we get the average delivery delay which is the average of the delay for all the delivered messages. The overhead of the protocol is defined as the ratio of total bytes transmitted during the simulation to the total bytes of actual data that have been delivered to their final destinations by the end of the simulation. Total number of bytes transmitted includes protocol bytes and data bytes and measures the total amount of bandwidth consumed by

DOI: 10.1002/sec

the protocol. The overhead indicates the cost incurred in terms of bytes transmitted while delivering each byte of the actual data that has been delivered to its final destination.

6.4. Experiment

Currently we only study the performance of the EnPassant protocols under ideal conditions of operation. In particular, we assume that the DTN nodes to have unlimited buffer capacity for holding in-transit data. Also the links have unlimited bandwidth and negligible latency. These conditions are not practically feasible but they allow us to get an unbiased evaluation of our protocols and validate our assumptions about their behavior. In the future, we will study EnPassant protocols under more realistic conditions.

For every combination of EnPassant protocol and underlying routing algorithm, we set the context by varying the number of pawns in the path as well as the total number of groups in the system. For each context, we run the simulation a total of five times, once for each of the five consecutive days mentioned in our scenario. We calculate the final values for the delivery ratio, average delivery delay and the protocol overhead as an appropriate average of these five sets of results. The delivery ratio is calculated as the percentage of total messages delivered during the five days out of the total messages injected into the system in that time. Average delivery delay is calculated as the weighted mean of average delays for the five days, weighing them by the number of messages delivered on the corresponding day. The protocol overhead is calculated as the ratio of total bytes transmitted in five days to the total data bytes delivered to their final destinations in that time. Since the privacy is measured based on path similarity from multiple runs in a single scenario, we only study the privacy of different protocols for a fixed setting of five groups and two pawns in the path.

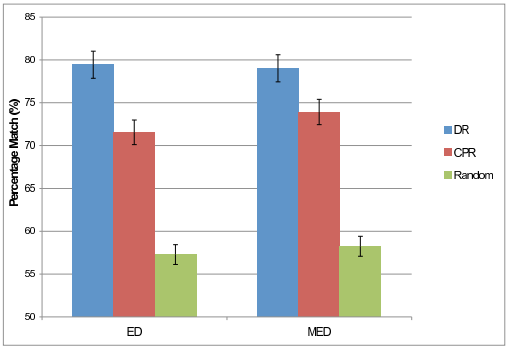

Fig. 2. **Path similarity (%) for different protocols.**

7. Simulation Results

In this section, we present the results of our simulations. DR/ED, CPR/ED and Random/ED are DR, CPR and Random Selection used with ED, respectively. Similarly, DR/MED, CPR/MED and Random/MED are DR, CPR and Random Selection used with MED, respectively.

7.1. Privacy

We begin by presenting a comparison of the privacy provided by the different protocols. We ran all experiments with five pawn groups and two pawns in each path. Figure 2 shows the path similarity for each

Prepared using secauth.cls
DOI: 10.1002/sec

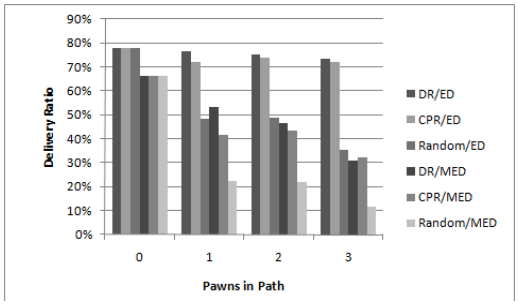

Fig. 3. **Delivery ratio for varying no. of pawns in the path.**

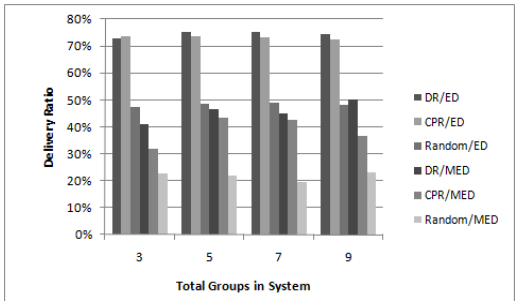

Fig. 4. **Delivery ratio for varying no. of groups.**

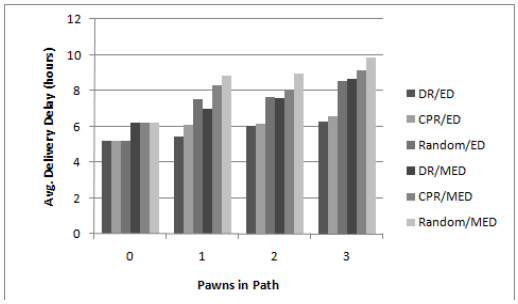

Fig. 5. **Delivery delay for varying no. of pawns.**

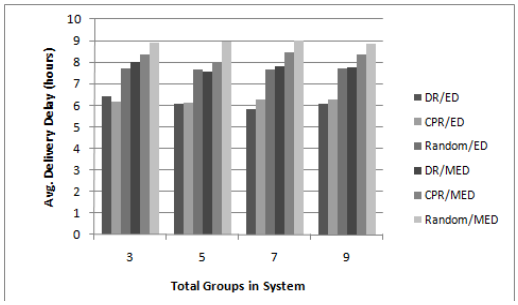

Fig. 6. **Delivery delay for varying no. of groups.**

combination of EnPassant protocol and underlying routing protocol. The path similarity is the average match percentage according to the Levenshtein edit

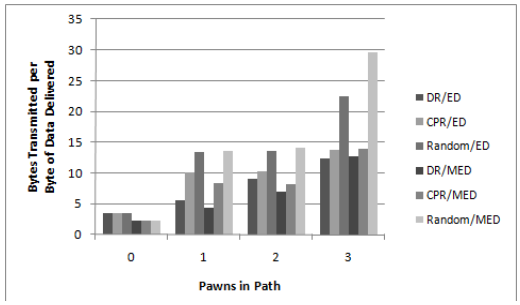

Fig. 7. **Overhead for varying no. of pawns in the path.**

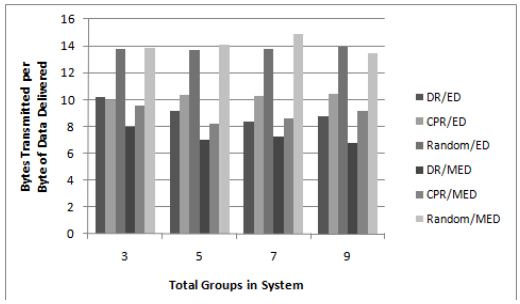

Fig. 8. **Overhead for varying no. of groups.**

distance, as described in Section 3.3. The higher the match, the lower the privacy, as more of the path is predictable. As expected, Random Selection provides better privacy than CPR, which in turn provides better privacy than DR. We see relatively little difference between the choice of ED and MED. Using ED as an example, the percentage match is 79.4% for DR, 71.6% for CPR, and 57.2% for Random Selection.

Based on the performance results we report below, this suggests that CPR is probably a better tradeoff between privacy and performance than DR. On the other hand, while Random Selection is clearly better in terms of privacy, it also have much worse performance.

Prepared using secauth.cls
DOI: 10.1002/sec

7.2. Performance of EnPassant Protocols

We now compare the performance of the proposed EnPassant protocols. The default number of groups in system is $g = 5$ and the default number of pawns in path is $p = 2$.

From Figures 3 and 4, we can see that over the same routing algorithm, the delivery ratio for random selection is much less than that of CPR. This is because firstly choosing a random pawn may result in a path which may not lead to the responder or incur very high delay so that the message can't be delivered on the same day. We also observe that the delivery ratio for DR is the highest because in DR the process of determination of the path is directed in such a way so as to most closely resemble the path computed by the underlying routing algorithm. However, the delivery ratio for DR is only slightly higher than that of CPR. The values of delivery ratios observed for Random/ED, CPR/ED, and DR/ED are 48.7%, 73.8%, and 75.1%, respectively.

The results for average delay are shown in Figures 5 and 6. As expected, for the same routing algorithm, the delay for random selection is the highest. Note, that the expected value of delay in random selection is the population mean of cost, in terms of delay, for the nodes in the next pawn group. CPR has less delay than random selection because if the minimum delay from current sender to next pawn is half of the population mean then the overall delay will also be reduced by half. Further, delay for DR is slightly less than that of CPR because we not only try to choose the pawns closer to the sender but we try to find them in the groups which are most likely to have nodes that are also closer to the responder. The values of average delay observed for Random/ED, CPR/ED, and DR/ED are 7.7 hours, 6.1 hours, and 6.0 hours, respectively. Although the delay values may seem high, as they are in hours, this is not unusual for DTN experiments [1]. In more connected scenarios, the delay values would be much more reasonable. Again notice that the average delay for Random Selection is considerably higher than that of CPR, whereas the average delay for CPR is only slightly higher than that of DR.

We observe a similar trend in the case of overhead. Figures 7 and 8 indicate that bytes transmitted per byte of data delivered is least for DR, slightly higher for CPR, and highest for random selection. The values of overhead for DR/ED, CPR/ED, and Random/ED are 9.1 bytes/byte, 10.3 bytes/byte, and 13.7 bytes/byte, respectively. In general, from our observations of delivery ratios and average delay we can say that the performance of CPR is much better than that of Random Selection and the performance of DR is slightly better than that of CPR.

7.3. Choice of Routing Algorithm

We now describe the effect of choice of underlying routing algorithm on the performance of EnPassant protocols. Let us fix the number of pawn groups to five and the number of pawns in each path to two.

DOI: 10.1002/sec

From Figures 3 and 4, we can infer that the delivery ratio for any EnPassant protocol when used with ED is much higher than when used with MED. For example, DR/ED has a delivery ratio of 75.1% which is much higher than 46.5% which is the delivery ratio observed for DR/MED.

From Figures 5 and 6, we see that the average delay for delivered messages is lower for an EnPassant protocol when the underlying protocol is ED rather than MED. For example, DR/ED has a lesser delay of 6.0 hours compared to the delay for DR/MED which is 7.6 hours.

However, from Figures 7 and 8 we see that when DR and CPR are used in combination with ED then they have a higher overhead than when they are used with MED. On comparing DR/ED and DR/MED, we see that the former has a higher overhead of 9.1 bytes/byte and the latter has a lower overhead of 7.0 bytes/byte. This trend is not observed in the case of random selection. The cause for the additional overhead is unclear; it may be that messages are being routed through more nodes based on more precise estimates of the available paths. Further investigation is required to learn more about how these routing protocols work.

In general, for any EnPassant protocol, the performance is better when the underlying routing algorithm has more accurate knowledge of the schedule, although a slightly higher overhead may be incurred.

7.4. Effect of Number of Pawns in Path

To study the effect of varying number of pawns in the path we consider the case where total number of pawn groups in the system is five.

Note that when the number of pawns in path is 0, EnPassant is not used and we instead observe the behavior of the underlying routing algorithm. We can see this case in Figures 3, 5, and 7 and see that for ED the value of delivery ratio is 77.9%, delay is 5.2 hours, and overhead is 3.5 bytes/byte. Similarly, we can also infer that for MED the value of delivery ratio is 66.3%, delay is 6.2 hours, and overhead is 2.3 bytes/byte.

Figure 3 indicates that in each of the six cases (DR/ED, CPR/ED, Random/ED, DR/MED, CPR/MED, and Random/MED), there is a slight decrease in delivery ratio as the number of pawns in path increases. For example, for DR/ED the delivery ratio gradually decreases from 77.9% to 73.4% as the number of pawns increase from zero to three. Similarly, for DR/MED the delivery ratio steadily decreases from 66.3% to 30.8% as the number of pawns increase from zero to three. As an exception, for CPR/ED, Random/ED, and CPR/MED we see that the delivery ratio is lesser when the number of pawns is two than when it is one. However, the difference between the corresponding delivery ratios in these cases is negligible (less than 1.7%) and hence the two values can be considered almost equal. In general, we can say that delivery ratio decreases with an increase in number of pawns.

From Figure 5, we can see that there is a gradual

DOI: 10.1002/sec

increase in average delivery delay with an increase in the number of pawns. For example, when the number of pawns increases from zero to three, the delay increases from 5.2 hours to 6.3 hours for DR/ED and it increases more substantially from 6.2 hours to 8.6 hours for DR/MED . Note that, for most combinations of EnPassant protocols and routing algorithms, there is a very small difference in delay values when number of pawns is one and when it is two.

In Figure 7, we can see that clearly more bytes will be transmitted per byte of data delivered as the number of pawns included in a path increases. Again if we consider the case of DR/ED and DR/MED, for the former the overhead increases from 3.5 bytes/byte to 12.3 bytes/byte, and for the latter the overhead increases from 2.3 bytes/byte to 12.7 bytes/byte.

Therefore, we infer that the performance will suffer if number of pawns is increased, as expected. However, since using two pawns provides much better protection than one pawn against a small group of malicious peers, it is worth using two pawns given the small increase in cost. Using three pawns instead of two is a more questionable tradeoff.

7.5. Effect of Group Size

Finally, we study the effect of group size in the system. For our data set, we use number of groups in the system between three and ten. From Figure 4, we can infer that the delivery ratio does not change with varying number of groups in the system. The values of delivery ratio remain more or less constant for all values of number of groups and this is true for all six cases of DR/ED, CPR/ED, Random/ED, DR/MED, CPR/MED, and Random/MED. Similarly, from Figures 6 and 8, we observe that average delivery delay and overhead, respectively, are also not affected by the number of groups in the system.

This means that large numbers of relatively small groups may be used to ensure greater protection from attackers, without much increase in the network costs.

8. Discussion

To better balance between routing performance and privacy, we propose Partially-Directed Routing (PDR). In PDR, the initiator first chooses minimum and maximum routing distances, according to the routing metric. The minimum distance should be selected to ensure a minimum privacy level, while the maximum distance should be selected to keep routing costs from going too high. PDR then identifies a set of paths through the network to the responder that satisfy the minimum and maximum distances and pass through nodes belonging to at least the minimum desired number of pawn groups. Note that finding all possible paths will often be prohibitively expensive to compute, but enough paths should be found to ensure that privacy is not harmed. One of the possible paths can be selected at random, possibly using weighted selection to improve either privacy or performance. The pawn groups will be selected based on which pawn groups are represented on the path. Finally, the routing is facilitated by indicating, at each layer of

DOI: 10.1002/sec

encryption, which member of the next pawn group should receive the message.

This approach may seem fragile, as it relies on the nodes expected to be present along the path. In the worst case, however, this is equivalent to random selection. Further, for PDR, we allow any node in the pawn group that gets the message to decrypt it, thereby improving the chances of finding a quick route. We can extend PDR with a flexible approach, Flexible PDR (FPDR), that combines PDR with CPR. In particular, each node can determine whether it expects to be able to reach a different node in the pawn group significantly faster than the pre-selected node. In this case, it modifies the pre-selected node to the faster choice. This can help the routing algorithm take advantage of changing conditions, although sometimes the new choice will be less than optimal when the entire route is considered. We will need to evaluate a threshold parameter that helps determine under which conditions the route should be changed.

PDR and FPDR allow for a fuller range of tradeoffs between privacy and performance. When the minimum distance is set to 0 and path selection is weighted for optimal performance, the algorithm becomes nearly the same as DR. In this case, the attacker has the greatest ability to take advantage of locality. A partial eavesdropper, for example, might be able to follow the direction of messages to identify possible initiators and possible responders. Against weak attack models, this approach should remain sufficiently secure, while providing near-optimal performance. To provide greater security, larger minimum distances and a more balanced weighting of the random path selection could be used.

9. Conclusions

In this paper, we proposed EnPassant, an architecture for privacy in message routing in opportunistic networks. The EnPassant design extends prior work in Internet anonymity with pawn groups, through which routing can happen more efficiently than with standard anonymity approaches based on random node selection. We described three ways of routing messages in the EnPassant framework and the trade-offs available in each method. We also described some of the cryptographic techniques required to make EnPassant work.

Finally, we describe the design and results of simulations based on traces of buses in the DieselNet framework. We showed that the DR protocol provides the best delivery ratio and delay, but CPR may provide the best trade-off of communication costs with privacy. Further, we found that the number of groups has little influence on the costs of our approach, while the number of intermediate nodes makes a big difference in all protocols.

Acknowledgements

This work was supported in part by the National Science Foundation under CAREER award number CNS-0954133 and award number CNS-0916221. Any

DOI: 10.1002/sec

opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

We must acknowledge that DTNSim2 and the DieselNet traces were most important for our simulation experiments. We would like to thank Jiangtao Li for his input on the cryptographic requirements of the system.

References

1. Jain S, Fall K, Patra R. Routing in a delay tolerant network. *Proc. ACM SIGCOMM*, 2004.
2. Jones EPC, Li L, Ward PAS. Practical routing in delay-tolerant networks. *Proc. ACM SIGCOMM*, 2005.
3. Kate A, Zaverucha GM, Hengartner U. Anonymity and security in delay tolerant networks. *Proc. SecureComm*, 2007.
4. Chaum D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* Feb 1981; **24**(2):84–88.
5. Dingledine R, Mathewson N, Syverson P. Tor: The next-generation Onion Router. *Proc. USENIX Security Symposium*, 2004.
6. Chaintreau A, Hui P, Crowcroft J, Diot C, Gass R, Scott J. Impact of human mobility on the design of opportunistic forwarding algorithms. *Proc. IEEE INFOCOM 2006*, 2006.
7. Pentland A, Fletcher R, Hasson A. Daknet: Rethinking connectivity in developing nations. *IEEE Computer* Jan 2004; **37**(1):78–83.
8. Scott K, Burleigh S. Request for comments: 5050 bundle protocol specification Nov 2007. URL `http://tools.ietf.org/html/rfc5050`.
9. Vahdat A, Becker D. Epidemic routing for partially-connected ad hoc networks. *Technical Report CS-2000-06*, Duke University Jul 2000.
10. Seth A, Keshav S. Practical security for disconnected nodes. *IEEE ICNP Workshop on Secure Network Protocols (NPSec)*, 2005.
11. Asokan N, Kostiainen K, Ginzboorg P, Ott J, Luo C. Applicability of identity-based cryptography for disruption-tolerant networking. *MobiOpp '07: Proc. Intl. MobiSys workshop on Mobile opportunistic networking*, 2007.
12. Symington S, Farrell S, Weiss H, Lovell P. Internet draft: Bundle security protocol specification Feb 2010. URL `http://tools.ietf.org/html/draft-irtf-dtnrg-bundle-security-`
13. Danezis G, Dingledine R, Mathewson N. Mixminion: Design of a type III anonymous remailer protocol. *Proc. 2003 IEEE Symposium on Security and Privacy*, 2003.
14. Clarke I, Sandberg O, Wiley B, Hong TW. Freenet: A distributed anonymous information storage and retrieval system. *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, 2000; 46–66.
15. Hopper N, Vasserman EY, Chan-Tin E. How much anonymity does network latency leak? *Proceedings of CCS 2007*, 2007.
16. Danezis G. Statistical disclosure attacks: Traffic confirmation in open environments. *Proc. of Security and Privacy in the Age of Uncertainty, (SEC)*, 2003; 421–426.
17. Mathewson N, Dingledine R. Practical traffic analysis: extending and resisting statistical disclosure. *Proc. Privacy Enhancing Technologies workshop (PET 2004)*, 2004.
18. Wright M, Adler M, Levine B, Shields C. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Transactions on Information and Systems Security (TISSEC)* 2004; **7**(4).
19. Wright M, Adler M, Levine BN, Shields C. Passive logging attacks against anonymous communications. *ACM Transactions on Information and Systems Security (TISSEC)* 2008; **11**(2).
20. Zhu Y, Bettati R. Compromising location privacy in wireless networks using sensors with limited information. *Proc. Intl. Conf. on Distributed Computing Systems (ICDCS 2007)*, 2007.
21. Mehta K, Liu D, Wright M. Location privacy in sensor networks against a global eavesdropper. *Proc. IEEE International Conference on Network Protocols (ICNP)*, 2007.
22. Murdoch SJ, Zieliński P. Sampled traffic analysis by internet-exchange-level adversaries. *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, 2007.

Prepared using secauth.cls
DOI: 10.1002/sec

23. Díaz C, Seys S, Claessens J, Preneel B. Towards measuring anonymity. *Proc. Privacy Enhancing Technologies workshop (PET)*, 2002.
24. Serjantov A, Danezis G. Towards an information theoretic metric for anonymity. *Proc. Privacy Enhancing Technologies Workshop (PET 2002)*, 2002.
25. Levenshtein VI. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* 1966; **10**(8).
26. Snader R, Borisov N. A tune-up for Tor: Improving security and performance in the Tor network. *Proc. Network and Distributed Security Symposium (NDSS '08)*, 2008.
27. Bauer K, McCoy D, Grunwald D, Kohno T, Sicker D. Low-resource routing attacks against TOR. *ACM WPES*, 2007.
28. Wright M, Adler M, Levine B, Shields C. Defending anonymous communications against passive logging attacks. *Proc. IEEE Sym. on Security and Privacy*, 2003.
29. Quercia D, Hailes S, Capra L. Mobirate: making mobile raters stick to their word. *UbiComp '08: Proc. Intl. Conference on Ubiquitous Computing*, 2008.
30. Jones EP, Li L, Schmidtke JK, Ward PA. Practical routing in delay-tolerant networks. *IEEE Transactions on Mobile Computing* Aug 2007; **6**(8):943–959.
31. Balasubramanianm A, Levine BN, Venkataramani A. Enabling interactive applications for hybrid networks. *Proc. ACM Mobicom*, 2008.

Prepared using secauth.cls
DOI: 10.1002/sec