DTN Research Group
Internet-Draft
Expires: January 4, 2008

S. Farrell
Trinity College Dublin
S. Symington
The MITRE Corporation
H. Weiss
P. Lovell
SPARTA, Inc.
July 3, 2007

Delay-Tolerant Networking Security Overview
draft-irtf-dtnrg-sec-overview-03

Status of this Memo

Copyright Notice

Abstract

This document provides an overview of the security requirements and mechanisms considered for delay tolerant networking security. It discusses the options for protecting such networks and describes reasons why specific security mechanisms were (or were not) chosen for the relevant protocols. The entire document is informative, given its purpose is mainly to document design decisions.

Table of Contents

1. Introduction

This section places this document in its current context as one of a series of DTN documents.

1.1. This document

This document is a product of the IRTF (http://www.irtf.org/) Delay Tolerant Networking Research Group (DTRNG) and is being discussed on the dtn-security mailing list. See the DRNRG site (http://www.dtnrg.org/) for details of the various DTN mailing lists.

The intent is for this document to present a snapshot of the security analysis which has been carried out in the DTNRG. The document is not normative in any sense but is intended to be a companion document which explains further the reasons for the design choices which are documented elsewhere. The discussion includes updates based upon experience gained during implementation of the security protocol.

The structure of the document is as follows:

We first present a selection of threats which were considered during the analysis carried out so far.

We then present some security requirements derived from that analysis or elsewhere.

We next present some of the design considerations which were applied during the design of the security mechanisms.

And we finally discuss some of the remaining open issues in DTN security.

Given that this is simply an informative snapshot document, none of the above are intended to be exhaustive, nor should other documents be criticized because something is mentioned here, but not countered there.

While this document is being prepared in parallel with the various protocol and security specifications, we will generally try not to refer to the specific fields used in those documents since the details may change and maintaining consistency at that level is not a goal here. Where we do refer to such details, of course, the specification documents are normative.

1.2. Background

The overall delay tolerant networking (DTN) architecture is described in [1]. A DTN is an overlay network on top of lower layer networks, which may vary from node to node. Some of these are challenged by limitations such as intermittent loss of connectivity, long or variable delay, asymmetric data rates, or high error rates. The purpose of a DTN protocol is to support interoperability across such potentially stressed lower layer networks. The DTN overlay network specifies a bundle protocol which is layered on top of a "convergence layer", which is itself on top of other lower layers. The DTN Bundle Protocol [2] describes the format of the messages (called bundles) passed between DTN bundle agents that participate in bundle communications to form the DTN store-and-forward overlay network.

The Bundle Security Protocol Specification [3] defines the integrity and confidentiality mechanisms for use with the Bundle protocol together with associated policy options.

Two other documents exists which are now somewhat outdated but remain worthwhile reading: A tutorial [4] about DTNs and an early DTN security model document [5].

There is also a related lower layer protocol specifically designed for very long delay environments, called the Licklider Transmission Protocol (LTP), [6] and which is also being developed by the same group. Even though the LTP protocol shares some security features [7] with the bundle protocol, we will mainly reference the bundle protocol here since its environment is much more complex and there is also a separate LTP motivation document [8].

In this document we may refer to "messages" to mean either a bundle from the bundle protocol or a segment from the LTP protocol. The context should make the meaning clear in each case.

2. Threats

This section describes some of the threats considered during the design process of the DTN security mechanisms. In this discussion,and throughout the document, we try to highlight DTN-specific aspects, assuming the reader is generally familiar with basic networking security concepts.

2.1. Non DTN node threats

The first set of threats considered were those coming from network elements which aren't directly part of the DTN. As an overlay network, bundles typically traverse multiple underlying network elements on each DTN "hop". Any vulnerability in the bundle protocol can be exploited at any of those network elements.

DTN security must take into account the usual range of such potential exploits (masquerading, bit flips, etc.) but in contrast to most network protocols, as an overlay protocol, the bundle protocol is possibly an easier target. In particular, if it is possible to insert new bundles at such lower-layer "hops", then DTN nodes have to be capable of countering such insertions by where possible, detecting and quickly deleting such spurious bundles.

Conversely, it is equally possible to take advantage of lower layer network security services, but this isn't be visible from the DTN layer, and requires coordinated administration in order to be really effective.

2.2. Resource consumption

Due to the resource-scarcity that characterizes DTNs, unauthorized access and use of DTN resources is a serious concern. Specifically, the following can consume DTN resources and be considered threats against a DTN infrastructure:

1. access by unauthorized entities,

2. unauthorized applications controlling the DTN infrastructure,

3. authorized applications sending bundles at a rate or class of service for which they lack permission,

4. unauthorised bundle content modification,

5. compromised network elements, be they DTN nodes or not.

In addition to these threats, DTN nodes can act to assist or amplify

such resource consuming behavior as follows:

1. forwarding bundles that were not sent by authorized DTN nodes

2. generating reports not originally requested (e.g. if a bundle has been modified).

3. not detecting unplanned replays or other mis-behaviors

2.3. Denial of service

In addition to the basic resource consumption threats mentioned above there is also a range of denial of service (DoS) attacks which must be considered in the DTN context. DTNs are in this respect, in more-or-less the same position as other MANETs so all the problems with secure routing in ad-hoc networks [9] exist for many DTNs too!

DoS attacks can be mounted at any layer, from physical to application. Generally when developing a new protocol we should attempt two things:-

- Make it hard to launch an "off-path" DoS attacks by making it hard to "guess" valid values for messages, e.g. through using random values instead of counters for identifying messages.

- Make it easier to withstand "on-path" DoS attacks by providing a way to choke-off DoS traffic, e.g. by changing to a mode of operation where only fresh, authenticated messages are accepted and all others are dropped.

In a DTN environment, the generally longer latencies involved will probably act to make DoS attempts more effective, so protocol developers and deployments should explicitly consider DoS at all times.

As with all networks, security mechanisms will themselves create new DoS opportunities. Therefore whatever services and mechanisms are defined for DTN security should explicitly consider DoS. For example, mechanisms which involve certificate status checking (via some protocol to a key server) based on received messages create new DoS opportunities since such lookups consume resources on both the receiving node and the key server.

2.4. Confidentiality and integrity

In addition to resource consuming threats, DTN applications can also be vulnerable to threats against confidentiality and integrity, such as:

1. falsifying a bundle's source,

2. changing the intended destination,

3. changing a bundle's control fields,

4. changing other block or payload fields,

5. replay of bundles

6. copying or disclosing bundle data as it passes.

2.5. Traffic storms

Since DTN protocols generate traffic as an artifact of other traffic, manipulation of bundle content, genuine, forged or modified, can be used to create unwanted traffic. In a DTN operating sufficiently "close to the wire", such traffic can have serious affects.

The Bundle Protocol includes various messages containing administrative records (e.g. bundle status reports, custody signals) produced in response to original traffic. The protocol specification, however, includes a constraint that the status report request flags must be zero on all bundles containing administrative records. Although a bundle that is not a custody signal or status report may cause the generation of custody signals and/or status reports, bundles that are themselves custody signals or status reports are not permitted to cause the generation of custody signals or status reports. This constraint is designed to prevent bundle storms and it does in the case when bundles can not be modified. If a DTN node (or other network element) could modify a "forwarding report" by resetting its "Application Data Unit is an Administrative Record" bundle flag and setting its "forwarding report" request flag, then this could cause the forwarding of an additional status reports, and so on. Traffic could continue to be generated in this manner for as long as the values of the bundle processing flags and the status report request flags can be modified. While the constraint prohibiting bundles containing administrative records from generating other bundles containing administrative records helps prevent traffic storms, it may be best to entirely remove some of status reporting capabilities from the Bundle Protocol.

2.6. Partial protection is just that.

Not all DTN nodes need to protect all parts of all bundles. Fr example, some DTN nodes won't be able to protect the integrity of the entire bundle including its payload. Reasons range from lack of computing power to application (or lower) layer protection mechanisms

already applying integrity. Alternatively, some DTN nodes may choose not to protect all parts of all bundles in order to permit reactive fragmentation.

There are also cases when bundle blocks will be modified in transit (e.g. the dictionary in the primary block), or a "via" block which captures the route a bundle has followed. As a result, integrity checking on anything more than a hop-by-hop basis becomes unwieldy other than for the payload.

So it is possible that some fields of a bundle are strongly protected whilst others are effectively unprotected. Whenever such a situation occurs, it will be still possible for network elements to use the bundle protocol as a communications channel, perhaps covert or perhaps overt. Where such misuse is a concern, the DTN should either use different security options which cover the fields of concern, or else administrators must ensure that the bundles only traverse lower layers where the probability of such misuse is sufficiently small.

3. Security Requirements

This section describes some of the high-priority DTN security requirements.

3.1. End-to-end-ish-ness

Traditionally, protocols tend to provide security services which are used either (or both) on a hop-by-hop or end-to-end basis. For DTN security though, we require that these services be usable also between nodes which are not endpoints but which can be in the middle of a route.

For example, if a sensor network employs lower layer security and has some gateway sensor node which is more capable and is periodically connected to the Internet, we may use DTN security services to protect messages between that gateway node and the other DTN sources and destinations on the Internet-side of the gateway. In the case of a confidentiality service, this is clearly useful since bundles which leave the sensor network could be encrypted (by the gateway node) for the final destination. In the case of say a software download, new code might be integrity protected from the origin to the gateway which is able to check some relevant white or black lists or use some other software authorisation scheme which cannot practically be used from a sensor node.

In order to define services which can be used in these ways we distinguish between the sender of a bundle and the security-sender for an application of one of these services. Similarly, we can distinguish between the bundle recipient and the security-recipient (or security-destination) for a given application of a security service. Basically, the security-sender is the DTN node that applied the security service, and the security-recipient (or security-destination) is the DTN node which is the target for the security service - say the node expected to decrypt or do integrity checking.

The extent to which the various security services can be combined for the same or different security senders and destinations needs to be made clear in the relevant protocol definition. However, this should be kept as simple as possible since unwanted complexity is highly likely to make a DTN harder to manage, and thereby less secure.

Experience with fragmentation issues has shown that there is good reason to distinguish (at the protocol field level) between uses of these services which are intended to be hop-by-hop (i.e. between this and the next DTN node), as opposed to end-to-end, at least for integrity checking. Equally, a protocol might not need to make this distinction and might only define e.g. one confidentiality service

which can be applied multiple times for a single bundle with different combinations of security-sender and security-recipient.

There is another example in which DTN security services differ from more "normal" network security services. (Indeed this mode of operation might be useful in non-DTNs too!). When a message is authenticated using a digital signature, in principle any network element on the path can do some checking of that signature. If the message contains sufficient information (the supposed signer's public key or a resolvable reference thereto) then any node can at least check the cryptographic correctness of the signature.

Although useful, this is typically insufficient to decide how to process the message, since in many environments basically anyone could insert a public key and a signature, producing a message which passes this test. In most real cases, there are some additional checks that the signer is authorised, either explicitly by checking that the signer's name or key is authorised for the purpose, or implicitly by using a PKI (e.g. via an extended key usage extension). It turns out that all practical ways to perform such authorisation checks are problematic in some DTN cases due either to the lack of an authorisation server (e.g. due to latency to/from from the verifier to the relevant authorisation server) or to restricted node capabilities, such as the case of a sensor node.

In such cases, it may be sensible for some "bastion" node along the route to do the authorisation check and then to (again explicitly or implicitly) attest that the authorisation test has passed. Subsequent nodes, may however, for either data integrity or accountability reasons wish to also validate the cryptographic correctness of the signature. The end result might be a mechanism whereby the message has a signature plus some meta-data which are fully processed by the "bastion" node, whereas the signature is only partly processed by all subsequent nodes. (Note: The role of the "security-destination" concept in such cases is not yet clear.)

These issues are not addressed by the current series of DTN specfications and it is likely that a new document [10] will be required to deal with them.

3.2. Confidentiality and integrity

Since most protocol designs use common elements to deal with all cryptographic based security services and mechanisms, they will all be dealt with in this section.

DTN protocols should provide a means to encrypt protocol elements so that messages in transit cannot practically be read. The bundle

protocol itself provides no confidentiality for the source or destination endpoint addresses, or any other endpoints included in the dictionary. This can be achieved using bundle-in-bundle encapsulation (BiB) if necessary.

Clearly, calling for a confidentiality service implies a need for key management. However, DTN key management remains an open issue, so we presently expect DTN protocols to support pre-shared-keys (and/or known irrevocable certificates).

Similarly, DTN protocols should provide a means to apply an integrity check to a bundle so that the identity of the security-sender can be established and changes in sensitive parts of the message can be detected. The bundle authentication block (BAB) and payload security block (PSB) have been specified to provide these services on a hop-by-hop and end-to-end basis respectively. Again, this implies a need for key management which is not met so far.

Clearly a protocol should allow a fairly flexible combination of applications of the confidentiality and integrity services, though hopefully disallowing insecure combinations (e.g. a plaintext signature which is out of scope of a confidentiality service allowing plaintext guesses to be verified).

These services should allow sensible combinations of a range of standard cryptographic algorithms to be used and should also allow changes to be made over time to the set of acceptable algorithms.

3.3. Policy based routing

Since the DTN, as a piece of infrastructure, may be quite fragile, we require protocols to be cautious regarding their consumption of network resources.

We require that DTN protocols and implementations support mechanisms for policy-based routing. In other words each DTN protocol specification should state the security-relevant policy variables upon which routing and forwarding decisions can be made. While this is still a little vague, the expectation is that each DTN specification should, in its security considerations text, say which security issues may exist which require a routing or forwarding policy decision to be made.

In particular, since forwarding even a single bundle will consume some network resources, every DTN node must implicitly incorporate some element of policy-based routing. We do not expect that every DTN node will be able to handle complex policy decisions. A DTN node can be programmed to forward all bundles received in a deterministic

manner, (e.g. flooding the bundle to all peers other than then one from which it was received). Even such a simple minded node is however, implicitly implementing a policy - in this case a simple flooding policy. So, though we require all nodes to implement some policy, that policy can be very simple.

Regardless of how simple or complex a node's support for policy based routing/forwarding might be, DTN implementers should document the relevant aspects of the implementation. In the absence of such documentation a node might be deployed in an inappropriate context, potentially damaging an entire network.

Some DTN nodes will however be on boundaries of various sorts, whether they be network topology related, administrative, networking technology related or simply a case where this node is the first which is capable of handling complex policy decisions. At one stage these nodes were termed security policy routers, and were considered to be "special" nodes. Our current view though, is that all nodes are in fact policy routers with some implementing policies which are more complex than others.

All nodes implement policy to some extent but not all will be security-aware. Setting a security-destination other than the bundle destination imposes a routing requirement which is expressed only in security extension blocks. Some nodes will be unable to process these and might route bundles to their destination bypassing the security-destination(s). The result would be that the bundle integrity cannot be verified or that the payload is unreadable because it had not been decrypted at the security-destination.

We do not, at this stage, require that there be an interoperable way to transfer policy settings between DTN nodes. Such a system could perhaps be developed (though it is an extremely complex task), but pragmatically, for now we consider the development of a DTN specific policy language and distribution framework out of scope.

DTNs themselves do not appear to generate many new types of policy based controls - the usual ingress, egress and forwarding types of control can all be applied in DTNs. For example, some "bastion" node might insist that all inbound bundles be authenticated and might add an authentication element to all outbound elements. So all the usual forms of control can, and should be, available for use in DTN nodes.

The DTN specific policy controls identified thus far, and for which we would recommend support include:

- time-to-live (TTL) type controls where we consider the amount of time for which a bundle has been "in-flight"

- controls to do with "strange" routes, such as those that loop

- controls handling local or global information about resource constraints in the DTN (e.g. knowledge of a peer's storage capacity)

- controls related to special types of fragmentation (e.g. reactive fragmentation) which are defined in a DTN

No doubt, more will be identified as DTN implementation and deployment experience is gained.

DTN node implementations will also be required to control access to whatever DTN interface they provide so that only authorized entities can act as the source (or destination) of bundles. Clearly this aspect of access control is an implementation, rather than a protocol issue.

It must be noted that policy based routing, if not deployed appropriately, may inadvertently create bundle "sinkholes". Consider the case in which a bundle is fragmented and one fragment of the bundle reaches a router whose policy requires it to see the entire bundle. All fragments of that bundle must also pass through that same router and, if they do not, then eventually the fragment at our paranoid router will expire. Ultimately the entire bundle never arrives at the intended destination. This is clearly a case to avoid - doing so, may be difficult to arrange without good route control.

4. Security Design considerations

This section discusses some of the security design considerations used during the development of the DTN security mechanisms

4.1. Only DTN-friendly schemes need apply

The high round-trip times and frequent and unpredictable disconnections that are characteristic of DTNs mean that security solutions which depend on ubiquitous online security services cannot generally be applied. Therefore solutions requiring ubiquitous access to servers (e.g. Kerberos, XKMS) are problematic. This is more-or-less analogous to the way that TCP won't work in DTNs. Such solutions might be usable from a range of DTN nodes within some security domain, although in that case what happens when a route spans more than one such domain remains to be researched.

The long delays that may be inherent in DTNs mean that data may be valid (even in-transit) for days or weeks, so depending on message expiration alone to rid the network of unwanted messages will also be problematic. How long is "too long" and how short is "too short", and how well are the clocks synchronized?

The impact of this design consideration most obviously applies to key management, but it will also apply to other aspects of security including distribution of new policy settings.

4.2. TLS is a good model

The Transport Layer Security (TLS) specification [11] provides us with some useful design ideas, especially in its use of "ciphersuites". In TLS, a ciphersuite is a single number that defines how all of the various cryptographic algorithms are to be used. The ciphersuite number is used in TLS negotiation. One of the more common ciphersuites is usually called "TLS_RSA_WITH_3DES_EDE_CBC_SHA" indicating that the TLS protocol is being used with RSA based key transport and with a variant of triple-DES as its bulk encryption algorithm and SHA-1 for various digesting tasks.

DTN can use a ciphersuite value in the same way - to indicate which cryptographic algorithms are in use for what purpose. This is how we can support both symmetric and asymmetric mechanisms for our cryptographic security services, and also allows us to extend DTN security in the future (e.g. use of identity based cryptography schemes).

In DTNs, we won't be doing negotiations of the sort done in TLS that

require multiple round-trips, but we can still use the ciphersuite idea. In fact, we extend it a little more to use the ciphersuite to also indicate which services are being applied (integrity or confidentiality) and also the set of input bits for the service. In this way we can distinguish between an integrity service which only protects the blocks from one that also protects the payload.

The DTN concept of ciphersuite also encompasses the idea of having different parts of the bundle protected by the relevant security service, as described in the next section.

4.3. Fragmentation

Fragmentation of a bundle is one of the more difficult DTN problems. There are many scenarios and what may work well for some may be useless for others. The two major issues are:

- some DTN networks may have extraordinary long propagation delays, which may look more like two one-way links

- the bundle payload is a single block and not a sequence of packets

The one-way-link issue is a severe handicap to the sending node, as it has little or no feedback on the progress or status of the transmission. Cooperation between the sender and receiver is difficult or impossible.

The payload is a single block but it can be split into smaller pieces as long as each becomes its own bundle, sometimes called a "fragment bundle". These are treated individualy after the fragmentation event and can be sent separately to the destination, and with varying levels of security depending upon the paths taken.

Fragmentation in DTN can happen in two ways. "Proactive fragmentation" is the deliberate action of a node, which has an entire bundle, to break it into smaller pieces. So-called "reactive fragmentation" is where we try to optimize retransmission after a connection failure of some kind. It assumes some level of interaction between the sender and receiver, so that the sender can restart from the point of failure or thereabouts. In this way, even very large bundles can be sent across intermittent or episodic links, piece by piece, and the fragments reassembled later.

Proactive fragmentation is reasonably interoperable with security processing but reactive fragmentation does not work well. As an example, consider the case of a node that has received the first 10 MB of a 20 MB bundle when the link fails and cannot be recovered by the underlying transport layer. The node has to decide whether to

forward the 10 MB fragment as a fragment-bundle. However, it cannot be integrity checked since not all bytes are present, which clearly is a breach of integrity.

The receiving node might request the sender to create and send a signature for the amount received, which would be faster than a complete bundle retransmission. The first fragment with its integrity check could be forwarded, and the original sender could create another fragment-bundle containing the remainder of the initial bundle data. This approach presupposes a high level of coordination, and also that a suitable link can be reestablished.

An alternative discussed for handling this is to associate a number of checksums with the bundle - say one for every 100k in this case, so that the entire bundle would use 20 checksums to provide end-to-end integrity. If the reactively forwarded fragment has the first 10 checksums its integrity can be checked. This comes at the expense of complexity and additional bytes of overhead (in this case perhaps 400 or more bytes), so it won't be desirable in most cases. Since each checksum protects a part of the payload, this scheme has been referred to as the "toilet paper" scheme - each forwardable fragment consisting of a number of sheets of payload-paper with its associated checksum. In order to support this type of fragmentation, we would have to define the relevant toilet paper ciphersuites in the security protocol specification. (At the time of writing, hopefully these schemes can be deprecated since they are clumsy and overly-complex for the benefit achieved.)

In summary the DTN concept of ciphersuite is borrowed from TLS and slightly extended to allow different parts of the bundle to be protected by the relevant security service. However, in general DTNs cannot support the use of the TLS handshake protocol as used in the terrestrial Internet.

One additional problem has recently become apparent and is currently under investigation. Various actions change the payload and the specific problem is that the payload length changes when encrypted using a block-mode cipher. This creates ambiguity for custody-transfer and for fragment-reassembly.

4.4. Naming and identities

Most security mechanisms work well with only certain kinds of identity. For example, Kerberos style security tends to go with domain-specific login names, PKI tends to work best with X.500 or LDAP-style names, and to a lesser extent with RFC822 addresses. In bundles, endpoint identifiers are represented as URIs. However, there is no well-defined URI scheme specifically required to be

supported. As a result, there is work to be done to map URIs to the types of identity which are easily supported by whatever mechanisms being considered.

In LTP, identities are flat octet strings, so again there is work to be done to map from these to e.g. user identities in a specific security mechanism.

4.5. Placement of checksums

As currently specified, the bundle protocol requires that the last block in the bundle be identified as such by setting its "Last block" block processing flag. This bit enables security blocks to be placed either before or after the bundle payload block. The ability to place a signature after the bundle payload block, at the end of the bundle, is important for integrity-protecting some bundles at nodes that have limited buffer space. For example, if a node wishes to sign a 10Mb bundle, but it only has 1Mb of usable buffer, then creating a digital signature over the 10Mb and sending that out before the end of the payload is simply impossible.

However, due to the properties of most hash functions, were the signature to be placed at the end of the bundle, then such a constrained node could in fact send out the signed bundle. This is due to the fact that hash functions have a continuation property which allows the to-be-hashed data to be fed through the function in blocks with only a small amount of state information required to be stored.

For this reason, DTN security protocols have the option of placing either a single block in the message or placing correlated blocks in the message, for example one at the start of the message which specifies the signature/hashing to be used (the ciphersuite in our case), and one (that contains the actual signature or MAC) at the end of the entire bundle. The ability to place such correlated security blocks in the message allows even very memory-constrained nodes to be able to process the bundle and verify its security result.

4.6. Hop-by-hop-ish-ness

In the above we discussed how security services can be applied which are not "truly" end-to-end. In the limit of course, we can use such a scheme to apply security just between this DTN node and the next hop. In particular there is clearly benefit in many cases from applying integrity checks on such a hop-by-hop basis.

There are two things worth noting about this particular case:

- Even though the protocol data units involved in end-to-end-ish and hop-by-hop-ish applications of security services may be almost identical, there may be benefit in artificially distinguishing between them since one could imagine many nodes which would only ever require (and thus properly support) hop-by-hop security. In fact, one could reasonably define ciphersuites which are only useful in such a hop-by-hop fashion.

- There doesn't seem to be much interest in making such an artificial distinction for confidentiality services, perhaps since the ability to use lower layer security is presumed to be much more common when the DTN nodes are "close" like this.

In any case, the current version of the bundle security protocol does use different blocks for hop-by-hop vs. end-to-end integrity.

5. Open Issues

This section discusses some of the issues which are still very open, either due to a lack of consensus in the DTNRG, or due to there being areas (like DTN key management) where much basic research remains to be done.

Where an issue has been discussed previously (e.g. source confidentiality), we will not include it here again.

5.1. Key management

The major open issue in DTN security is the lack of a delay-tolerant method for key management. We are at the stage where we only really know how to use existing schemes, which ultimately require an on-line status checking service or key distribution service which is not practical in a high delay or highly disrupted environment.

Note that even though some identity based cryptography (IBC) schemes superficially appear to solve this problem (once we assume that the originator has a name for the destination endpoint), this is in fact not the case. The problem is that current IBC schemes effectively act only as a kind of "group certificate" where all of the nodes using a given private key generator can use a single "certificate", but the problem of validity for that "certificate" (which will contain the generator's parameters) is the same problem as verifying a CA certificate in a standard PKI.

So, the only generally applicable schemes we currently have are basically equivalent to shared secrets or else irrevocable public key (or certificate based) schemes. Clearly, this is an area where more research work could produce interesting results.

5.2. Handling replays

In most networking scenarios, we either wish to eliminate or else dramatically reduce the probability of messages being replayed. In some DTN contexts this will also be the case - particularly as replaying a (e.g. authenticated, authorized) message can be a fairly straightforward way to consume scarce network resources.

However, there are also DTN scenarios where we wish to deliberately replay messages, even to the extent of routing messages around a loop. For example, if Bob is willing to act as a data mule for Alice, who has limited storage, then Bob might pick up a bundle as he passes Alice on his outbound journey from his Internet-connected home location. As he goes on however, Bob also runs into storage problems, so he temporarily deposits the bundle with Charlie, who

he's passing now, and who he'll also pass on his way back home, in say, a week's time. After a week, Bob indeed passes Charlie again and picks up that bundle for the second time, after which he goes on to successfully deliver the bundle via the Internet-connected node at home. Now in this scenario, the same bundle is received by Bob twice, and so would likely trigger any replay detection algorithm that Bob is running, but of course, the behavior as described is nominal for the circumstances presented.

In addition, there are some routing schemes which involve duplicating messages. For example, a node might flood all its peers with a copy of a message to increase the probability that it will arrive at the destination before it expires. Clearly such routing schemes are likely to result in nodes seeing the same message more than once, but it's not clear whether any such node would be correct to delete such apparent "duplicates".

The element of delay in DTNs also complicates handling replays. Replay detection schemes generally depend on noting some unique aspect of messages (via digesting of some message fields) and then keeping a list of (the digests of) recently seen messages. The problem in the DTN context is the "recently seen" part of such replay detection algorithms, since maintaining a list for say 30 days would be fairly resource intensive, but might be required if latencies are of that size. So the most obvious ways to protect against replays are problematic.

The result is that the extent to which we can, or should, define a generic DTN replay detection scheme is hard to determine and at this point remains an open DTN security issue. It may be that this means that schemes need to be specified as part of a bundle routing algorithm.

One aspect of replay handling where security can be enforced is setting the final destination bundle node to deliver each bundle only once to its application. In this way, even though replays can consume network resources, they are less likely cause application layer damage. An example of such damage would be a protocol which used a bundle to represent "Move the telescope 10 degrees left" - repeated replays of this message could result in damage if the telescope is pointed at the Sun. Of course, the application layer in this case ought also be detecting replays, e.g. by including a command number, but the example does demonstrate the point.

Additional discussion relevant to at-most-once-delivery and a way to handle intentional resending of a bundle can be found in the DTN Retransmission Block specification [12].

5.3. Traffic analysis

We do not currently define any security services for protecting against traffic analysis. A general traffic analysis protection scheme is probably not, in any case, a realistic goal for DTNs, given their tendency to be resource-scarce and there have been no calls for a generic approach to this problem. However, for some disruption tolerant networks, hiding traffic (e.g. the existence of a signal from a sensor net) may be a very important security requirement.

So, the first open issue here is the extent to which there is a real need for a generic scheme for protection against traffic analysis. If there were, then the second open issue is how to define such a scheme to be delay and disruption tolerant and which also doesn't consume too many resources.

Finally, traffic analysis protection may be left as a local matter for the underlying network layers, e.g. if a particular radio link were of concern, then total obscuration of that link may be required, and may in fact be the only way to hide such radio traffic.

5.4. Routing protocol security

Clearly whenever DTN routing protocols are defined they will introduce new security requirements, or at least change the emphasis to be properly placed on meeting the various requirements posited above. For example, one could expect that a robust and scalable origin-authentication scheme would become more important.

At the time of writing there are no well-documented DTN routing protocols, so DTN routing protocol security must clearly be in our list of open issues. However, if a putative DTN routing protocol were to use either the Bundle protocol or LTP, it could clearly make use of their existing security features.

5.5. Multicast security

In a DTN, bundles are sent to destination endpoints, and any given endpoint consists of a set of zero or more bundle nodes. A bundle that is sent to a given destination endpoint must be sent to all of the nodes that are in the minimum reception group of that endpoint. If an endpoint may contain multiple nodes and its minimum reception group is all of the nodes registered in that endpoint, then a bundle sent to that endpoint is functionally similar to "multicast" operations in the Internet. If an endpoint may contain multiple nodes and its minimum reception group is any given number of the nodes registered in that endpoint, then a bundle sent to that endpoint is functionally similar to "anycast" operations in the

Internet. Extensions to the bundle protocol for providing custodial transfer of bundles sent to multicast endpoints have been defined in [13].

Within DTN, there is currently no mechanism defined for restricting which nodes may register in a "multicast" or "anycast" endpoint. The security architecture currently does not address the security aspects of enabling a node to register with a particular mutlicast or anycast EID. Without a capability to restrict the registration of nodes in multicast or anycast endpoints, any node may register in such an endpoint and thereby receive traffic sent to that endpoint. In addition, even though an endpoint may be a singleton endpoint, meaning that it is not permitted to contain more than one node, it may be possible for a second (or more) node to register in a singleton endpoint and receive bundles that are sent to that endpoint if the bundles are routed in such a way that they are forwarded to that node (e.g. using flood routing).

The mandatory end-to-end(ish) confidentiality and authentication ciphersuites that are defined in the Bundle Security Protocol require that all destination nodes use the same key material to decrypt and authenticate the received bundle. Modifications to the mandatory end-to-end(ish) ciphersuites or additional ciphersuites would need to be defined to provide the possibility that a bundle could be encrypted or authenticated differently for different nodes in its multicast or anycast endpoint.

In addition, there are some new aspects to multicast endpoint membership security, given that most work to date has implicitly assumed that the signaling traffic (e.g. registering in the multicast endpoint) can occur in more-or-less "real" time. In a DTN, registering in a multicast endpoint may be more akin to signing up to a mailing list, so that bundles that originated before the registration occurred may be received afterwards. In principle, such a late registering node might get sent the entire mailing list archive either by design or in error. Even if some sort of mechanism to authenticate registering nodes were to be defined, there are still issues that arise out of the fact that the endpoint registration process may itself be lengthy. For example, if a registering node authenticates with some credential that has a notBefore time of January 1, 2007 (as an X.509 public key certificate might have), and some bundle created before that time is still to be delivered, should the notBefore time of the credential be part of the decision as to whether to route a given bundle to the registering node? In this case, probably the answer is "no", but in some contexts that could be the wrong answer, allowing new (cheap) identities access to old (expensively accrued) materials.

5.6. Performance Issues

Provision of security within a DTN imposes both bandwidth utilization costs on the DTN links and computational costs on the DTN nodes.

The provision of DTN security will consume additional bandwidth. The amount consumed depends on the way optional parameters are encoded, or not, and on thecryptographic algorithms used. In addition, if more than one security service is used for the same bundle (e.g. a MAC to be removed by the next hop and a signature for the final destination) more of the possibly limited amount of bandwidth available for security purposes will be used.

The use of DTN security also imposes computational costs on DTN nodes. There may be limits regarding how much CPU can be devoted to security and the amount of computation will depend on the algorithms used and their parameters.

6. Security Considerations

Since this entire document is an informative description of how the DTNRG are approaching security, there is little to say in this section.

However, implementers of DTN protocols must not take text here to be normative, in the case of conflict the relevant protocol specification takes precedence.

7. IANA Considerations

None.

8. Informative References

[1] Cerf, V., Burleigh, S., Durst, R., Fall, K., Hooke, A., Scott, K., Torgerson, L., and H. Weiss, "Delay-Tolerant Network Architecture", RFC 4838, April 2007.

[2] Scott, K. and S. Burleigh, "Bundle Protocol Specification", draft-irtf-dtnrg-bundle-spec-09.txt, work-in-progress, April 2007.

[3] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", draft-irtf-dtnrg-bundle-security-03.txt, work-in-progress, April 2007.

[4] Warthman, F., "Delay-Tolerant Networks (DTNs) A Tutorial", http://www.dtnrg.org/ , March 2003.

[5] Durst, R., "An Infrastructure Security Model for Delay Tolerant Networks", http://www.dtnrg.org/ , July 2002.

[6] Ramadas, M., Burleigh, S., and S. Farrell, "Licklider Transmission Protocol", draft-irtf-dtnrg-ltp-06.txt, work-in-progress, October 2007.

[7] Farrell, S., Ramadas, M., and S. Burleigh, "Licklider Transmission Protocol - Extensions", draft-irtf-dtnrg-ltp-extensions-05.txt, work-in-progress, October 2007.

[8] Burleigh, S., Ramadas, M., and S. Farrell, "Licklider Transmission Protocol - Motivation", draft-irtf-dtnrg-ltp-motivation-04.txt, work-in-progress, October 2007.

[9] Zhou, L. and Z. Haas, "Securing Ad-Hoc Networks", IEEE network vol 13, no. 6, Nov-Dec 1999, pp 24-30.

[10] Farrell, S., "DTN Authentication, Authorization and Accounting", draft-irtf-dtnrg-bundle-aaa.-00.txt, work-in-progress.

[11] Dierks, T. and E. Rescorla, "The TLS Protocol - Version 1.1", RFC 4346 , April 2006.

[12] Symington, S., "Delay-Tolerant Network Retransmission Block", draft-irtf-dtnrg-bundle-retrans-00.txt, work-in-progress, April 2007.

[13] Symington, S., "Delay-Tolerant Network Multicast Custodial Transfer", draft-irtf-dtnrg-bundle-multicast-custodial-00.txt, work-in-progress, May 2007.

Authors' Addresses

Stephen Farrell
Trinity College Dublin
Distributed Systems Group
Department of Computer Science
Trinity College
Dublin
Ireland

Phone: +353-1-608-1539
Email: stephen.farrell@cs.tcd.ie

Susan Flynn Symington
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102
US

Phone: 703 983 7209
Email: susan@mitre.org
URI: http://mitre.org/

Howard Weiss
SPARTA, Inc.
7110 Samuel Morse Drive
Columbia, MD 21046
US

Phone: +1-443-430-8089
Email: hsw@sparta.com

Peter Lovell
SPARTA, Inc.
7110 Samuel Morse Drive
Columbia, MD 21046
US

Phone: +1-443-430-8052
Email: peter.lovell@sparta.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).