Network Working Group K. Viswanathan
Internet-Draft F. Templin
Intended status: Standards Track Boeing Research & Technology
Expires: September 9, 2016 March 8, 2016

Architecture for a Delay-and-Disruption Tolerant Public-Key Distribution Network (PKDN)

draft-viswanathan-dtn-pkdn-00.txt

Abstract

Delay/Disruption Tolerant Networking (DTN) introduces a network model in which communications can be subject to long delays and/or intermittent connectivity. DTN specifies the use of public-key cryptography to secure the confidentiality and integrity of messages in transit. The use of public-key cryptography posits the need for certification of public keys and revocation of certificates. This document formally defines the DTN key management problem and then provides a high-level design solution for delay and disruption tolerant distribution and revocation of public-key certificates along with relevant design options and recommendations for design choices.

Table of Contents

1. Introduction

The interactions in a public-key management system are between: (a) the sender and the receiver; and, (b) the receiver/sender and a trusted authority (Certificate Authority or CA). Although there are public key management systems without any trusted authority, like PGP and block-chain based certification, revocation of public keys in such systems are either impossible or complex. The certification process in such systems usually require many to and fro message transmissions, which is not suitable for delay and disruption tolerant conditions. For these reasons, the subsequent discussions in this document shall assume a trusted authority.

In any public-key cryptographic system, the sender must have an authentic copy of the receiver's public key for sending confidential communications. The receiver must have an authentic copy of the sender’s public key for receiving authentic communications. Key management protocols have required the sender/receiver to interact in near-real-time with the trusted authority to determine if a public key certificate has not been revoked. Such handshake communications usually use TCP [RFC0793]. But, near-real-time messaging is not feasible on DTN. Therefore, terrestrial key management protocols may not always function as intended on DTN.

The Online Certificate Status Protocol (OCSP) [RFC6960], for example, requires the receiver of a public key certificate to have on-demand interactions with a Certification Authority (CA) in order to get the current status information for the certificate. Three status responses may be received by the receiver from the CA, namely: good, revoked, and unknown. The receiver needs to accept good certificates and reject revoked certificates. The CA sends a response indicating the unknown state usually when it does not recognize the issuer of the certificate. In this case, the receiver is expected to interact on-demand with other CAs for determining if the certificate was revoked. When the status in the response is good, since the CA does not remember the receiver’s interest in the certificate, the receiver is required to periodically request the status before every use of the certificate.

OCSP is a resource intensive protocol. In order to reduce the round-trip costs for the temporal validation of the certificates, especially in constrained clients (receivers), a provision in TLS Extensions (see Section 8) [RFC6066] has been proposed so that the senders shall send what is called a "stapled Certificate Status" to the receivers. The stapled Certificate Status is a time-stamped certificate-status certificate obtained from a trusted authority by the sender. If the constrained receiver (client) accepts the stapled Certificate Status, then it need not interact with any CA to ascertain the temporal validity of the certificate -- thus reducing communication costs on the receiver side. Although such proposals are useful when dealing with constrained clients (or receivers of certificate), they only transfer the burden of certificate-status queries towards the senders and away from the receivers. Such mechanisms do not obviate the need for on-demand interactions.

The Secure/Multi-purpose Internet Mail Extensions (S/MIME) [RFC5751] allows a sender to encapsulate its certificate as a meta-data (in the message header) for processing an email message. The receiver is expected to consult with a Certificate Revocation List (CRL) or other certificate status verification mechanisms to validate the temporal validity of the certificate. Thus, S/MIME does not obviate the need for on-demand interactions with remote trusted authorities.

As mentioned earlier, on-demand interactions with any party, trusted or otherwise, is not feasible in the network model for DTN. Therefore, existing terrestrial key management protocols are not suitable for DTN. This proposal describes the high-level design choices for a mechanism, which can satisfy the requirements for DTN Key Management [I-D.templin-dtnskmreq], that does not require on-demand interactions with remote parties.

1.1. Related Documents

The following documents provide the necessary context for the high-level design described in this document.

RFC 4838 [RFC4838] describes the architecture for DTN and is titled, "Delay-Tolerant Networking Architecture." That document provides a high-level overview of DTN architecture and the decisions that underpin the DTN architecture.

RFC 5050 [RFC5050] describes the protocol and message formats for DTN and is titled, "Bundle Protocol Specification." That document provides details for the protocol message format for DTN, which is called as Bundle, along with the description of processes for generating, sending, forwarding, and receiving Bundles. It also specifies an encoding format called SDNV (Self-Delimiting Numeric Values) for use in DTN.

RFC 6257 [RFC6257] is titled, "Bundle Security Protocol Specification." It specifies the message formats and processing rules for providing three types of security services to bundles, namely: confidentiality, integrity, and authentication. It does not specify mechanisms for key management. Rather, it assumes that cryptographic keys are somehow in place and then specifies how the keys shall be used to provide the security services. Additionally, it attempts to standardize the cipher suite in DTN.

5050bis [I-D.ietf-dtn-bpbis] is an Internet Draft on standards track that intends to update RFC 5050. It introduces a new concept called "node ID" and relates it with an existing concept called "endpoint ID." A DTN endpoint is envisioned to contain one or more nodes. It also excludes extension blocks defined in RFC 5050 to be external to the primary block, which makes the primary block immutable by intermediary nodes. Thus, in 5050bis it is allowed that a node receives the primary block with extension blocks but without the capability to process the extension blocks. In the Security Considerations section, 5050bis explicitly describes end-to-end security using Bundle-Integrity-Block (BIB) and Bundle-Confidentiality-Block (BCB). It does not specify link-by-link security considerations to be part of the bundle protocol level using the Bundle-Authenticity-Block (BAB), which was described in RFC 6257. The convergence layers may provide link-by-link authentication instead of bundle protocol agent.

The Internet Draft [I-D.ietf-dtn-bpsec] for DTN communication security is titled, "Streamlined Bundle Security Protocol Specification (SBSP)." When compared with RFC 6257, it is silent on concepts such as Security Regions, at-most-once-delivery option, and cipher suite specification. It provides more detailed specification for bundle canonicalization and rules for processing bundles received from other nodes. Like RFC 6257, the draft does not describe any key management mechanisms for DTN but assumes that suitable key management mechanism shall be in place.

The Internet Draft for specifying requirements for DTN Key Management [I-D.templin-dtnskmreq] is titled, "DTN Security Key Management - Requirements and Design." It sketches nine requirements and four design criteria for DTN Key Management system. The last two requirements are the need to support revocation in a delay tolerant manner. It also specifies the requirements for avoiding single points of failure and opportunities for the presence of multiple key management authorities.

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT","SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. Lower case uses of these words are not to be interpreted as carrying RFC2119 significance.

This draft uses the following terminologies.

Sender
: has a public-key certificate. It must pass a message to the receiver via a validator in order to install its public-key certificate on the receiver.

Receiver
: receives messages from senders via validators and stores the sender’s public-key certificate, if the certificate is valid and has not been revoked.

Validator
: provides store-validate-and-forward service for public-key certificates from the sender to designated receivers. Additionally, it pushes revocation updates to the receivers for

public-key certificates, which were previously forwarded to the receivers by that validator.

Client
: consumes the services of the validator. Client must include the logic for sender and receiver. Therefore, a client must be able to send its certificates to others or receive certificates from other clients via the validator. The client must be able to receive revocation updates from validators.

Certificate Revocation Manager (CRM)
: is a trust authority that sends signed revocation notices to the validators so as to revoke public-key certificates.

Public Key Distribution Network (PKDN)
: is a strict DTN overlay network that acts as:

1. a store-validate-and-forward communication medium for communications from the senders to the receiver via the validators; and,

2. a multicast communication medium for communications from the CRM to the validators. The communication medium can be implemented using either DTN multicast communications or application-level message-propagation networks using recursive publish-subscribe relationships.

2. DTN Key Management

This section shall introduce the problem statement for DTN Key Management problem followed by an enumeration of communication-patterns that can be used for potential solutions and a proposed solution for the problem that is called a Public-Key Distribution Network.

2.1. The DTN-Key-Management Problem Statement

The problem of DTN Key Management can be visualized as shown in Figure 1. The Receiver receives a public key certificate from the Sender. Since the Sender is not trusted to share timely revocation information, the Receiver needs to receive timely revocation information from a Trusted Authority. A basic problem is: (a) how can the Trusted Authority know when the Receiver needs the revocation information for a Public-Key Certificate; and, (b) how can periodic and consistent revocation information be availability in timely and delay-and-disruption tolerant manner? The second question gains importance in DTN because the delay and disruption in the communication path between the Sender and Receiver may not be correlatable with that between the Receiver and the Trusted Authority. This makes the DTN Key Management problem different from terrestrial key management systems, where communication paths are assumed to be uniform, interactive, on-demand, and similar.

```
+---------+  Revocation     +--------+   Public-Key    +-------+
|         |  Information    |        |   Certificate   |       |
| Trusted |--(disruption/-->|Receiver|<--(disruption/--|Sender |
|Authority|  delay)         |        |   delay)        |       |
+---------+                 +--------+                 +-------+
```

Figure 1: DTN Key Management Problem

An analysis of the above problem using CAP theorem [CAP] suggests that when network partition occurs, due to delay or disruption, the receiver needs to make a local decision in favour of either availability of its service for the received message or consistency of its operations in not accepting revoked certificate, which was used to provide integrity service to the received message. In other words, when the Receiver has received the public key certificate but has not received any revocation information as yet, it needs to vote in favour of either: (a) availability, by accepting the certificate without waiting for revocation information; or, (b) consistency, by waiting for the receipt of revocation information. If it votes in favour of availability, it risks the use of inconsistent information. If it votes in favour of consistency, it risks lack of availability of the public-key for some dependent information processing, which must be paused. Clearly, in the presence of delay and disruption, both consistency and availability cannot be achieved.

DTN Key Management solutions must be partition tolerant and provide trade-off options for their applications between availability and security consistency. Such a trade-off may be realized in an application-agnostic manner by aiming for eventual consistency instead of immediate consistency. Eventual consistency means that all DTN nodes will eventually reject revoked keys but until such an eventuality some DTN nodes are allowed to work with stale revocation information depending on their application security sensitivity. Immediate consistency is not possible in DTN but is possible in the terrestrial Internet. The time available for accepting or rejecting the certificate (and the message) will be decided by the application’s security threshold.

2.2. Communication patterns for solving the DTN problem

As mentioned previously, the two-fold problem of DTN Key Management Problem is:(a) how can the Trusted Authority know when the Receiver needs the revocation information for a Public-Key Certificate; and,

(b) how can periodic and consistent revocation information be made available in timely and delay-and-disruption tolerant manner?

Five communication patterns can provide solutions to the first question (Question a), namely:

Pattern 1: (Request-response) The Receiver informs the Trusted Authority every time when it needs fresh revocation information for a certificate by sending a request. The Trust Authority responds with a fresh status information for that certificate.

Pattern 2: (Publish-subscribe) The Receiver informs the Trusted Authority about its interest in a certificate only once, which is the first time when it needs the revocation information, by sending a subscription request. The Trusted Authority responds to the subscription request with a fresh status information for that certificate and remembers the subscription request. Whenever there is a change in status information, the Trusted Authority sends the updates to the Receiver without having to receive a request for the same.

Pattern 3: (Blacklist broadcast) The Trusted Authority does not receive any certificate-specific request from any Receiver. It periodically broadcasts Certificate Revocation Lists (CRLs)to all DTN nodes including the Receiver. If the broadcast mechanism were to be replaced with a multicast mechanism, then the Receiver will be expected to register its address with the Trusted Authority exactly once as a registration process. Note that the registration process does not reference any certificate unlike the subscription process in the previous pattern.

Pattern 4: (White-list broadcast) This communication pattern is similar to the previous communication pattern except that the Trusted Authority periodically broadcasts a list of valid certificates instead of broadcasting a list of invalidated certificates. This communication pattern is useful when the number of certified public-keys are less.

Pattern 5: (Publish with proxy subscribe) The Sender sends its certificate through the Trusted Authority to the Receiver, who shall accept certificates only from the Trusted Authority. The Trusted Authority validates the certificate before forwarding it to the Receiver. The Trusted Authority subscribes the Receiver for interest in the Sender’s certificate so that periodic updates can be sent in the future for the certificate. Thus, the Sender acts as a proxy for the Receiver and subscribes the Receiver for future updates from the Trusted Authority.

Pattern 1 describes the communication style used by terrestrial key management solutions such as OCSP. The Receiver may receive the certificate from the Sender every time a security session is established as is the case in TLS [RFC5246]. Thus, the Receiver may need to send a request to the Trusted Authority every time a security session is established. Section 1 discussed why this communication style is not suitable for DTN.

Pattern 2 has a similar complexity as Pattern 1 for the first round of communication for a certificate between the Receiver and the Trusted Authority. The communication complexity greatly eases from the second round onwards when the Trusted Authority can send updates to the Receiver without requiring a request. Although this pattern improves the communication complexity from the second round onwards, it does not improve communication complexity of the first round of communications, which is a bottleneck in the DTN settings as described for Pattern 1 in Section 1.

Patterns 3 and 4 require periodical broadcast/multicast of a list data structure (CRL or list of valid public keys). The efficiency of such patterns depend on three factors, namely: the size of the list of revoked certificates, the number of communication recipients, and the frequency of communication. If any one of these factor were to increase, bandwidth utilization will be inefficient because not all recipients of the communication may be interested in all elements of the list that they receive. Thus, most recipients will end up discarding many communications that they receive from the Trusted Authority. When two or more of the factors were to increase simultaneously, the communication system may be overloaded and normal application communications may be affected. Clearly, this solution is not scalable with the increase in number of recipients. Additionally, since Pattern 4 uses white-lists and, in public key management, white-lists grow more frequently than black-lists, the frequency of communications between the Trusted Authority and the Receivers will be higher than in Pattern 3. Also, since the Receivers depend on the Trusted Authority for timely delivery of white-listed keys, the first communication from the Sender to the Receiver must strictly happen after the Trusted Authority has sent the Sender’s public key to the Receiver in a white-list communication. Otherwise, the Sender’s communication will have to be rejected by the Receiver even though the Sender may be in possession of a registered (or authorized) public key. This calls for increased out-of-band delay-tolerant synchronization between the Sender and the

Receiver. For reasons mentioned above, this document shall not pursue Patterns 3 and 4.

Pattern 5 requires every Sender to send their public-key certificates through the Trusted Authority to the Receiver. The Trusted Authority can be a validator, which is allowed to filter communications with revoked public-key certificates. Additionally, the validator remembers the Receiver's interest in order to send periodic revocation updates for the forwarded public-key certificates. The rest of this document shall employ this communication pattern.

3. PKDN Architecture

As mentioned in the previous section, this proposal adopts Communication Pattern 5 for designing Public Key Distribution Network (PKDN). The elements of PKDN and simplified information flow are shown in Figure 2. The sender sends its certificate, along with other information such as receiver's address, to a validator in the PKDN. The validator forwards valid certificates to the receiver and sends certificate revocation information to the receiver when such information is available. In order to make the information flow practical, addressing, timing, and security meta-data are sent along with the certificate, validated certificate, and certificate status. The details of the meta-data shall be described in the rest of this section.

```
                         +----------------------+
                         |Delay Tolerant Network|
                         |      +-------+       |
                         |      |  CRM  |       |
                         |      +---+---+       |
                         |          |           |
                         |          |Delta-CRL  |Validated
      +------+           |    +-----v------+    |Certificate +--------+
      |      |Certificate|    |   (PKDN)   +----------------->        |
      |Sender+----------------> Network of |    |            |Receiver|
      |      |           |    | Validators +----------------->        |
      +------+           |    +------------+    |Certificate +--------+
                         +----------------------+Status
```

Figure 2: Simplified view of PKDN Architecture

Figure 3 presents a simplified communication stack view of the same PKDN architecture (Figure 2). It does not depict the complete Bundle Protocol layering architecture for the sake of clarity and brevity -- RFC 5050 [RFC5050] contains the complete Bundle protocol layering architecture. All architectural elements of PKDN use the Bundle

Protocol (BP) layer as their communication interface. Thus, every PKDN architectural element is a Bundle Protocol Application.

```
                          +--------+
          +-----------+   |  CRM   | +-----------+
          |           |   +--------+ |           |
          | Validator |   |   BP   | | Validator |
          +-----------+   +---+----+ +-----------+
          |   BP      +-+     |   +--+   BP      |
          +-----------+ |     |   |  +-----------+
                        |     |   |
      +--------+     _( +===- |   |        +----------+
      |        |    (        -+ )-+- -_    |          |
      | Sender |   (  Delay Tolerant  _)   | Receiver |
      +--------+  (_    Network    _=-     +----------+
      |   BP   +----+    Cloud    _)       |    BP    |
      +--------+     -=__(__  _-)-         +-----+----+
                           -+-                   |
                            |                    |
                            +--------------------+
```

Figure 3: PKDN Architecture: Simplified communication stack view

3.1. PKDN Architectural elements

The architectural elements and their roles are as follows.

1. (Revocation Manager - RM) This element is the revocation authority for a PKDN. There can be one or more RMs in a PKDN. A revocation manager has a self-signed public key. The self-signed public key must be made available securely to all other architectural entities as an out-of-bound, single-time configuration.

2. (Sender) The sender of a message with a valid certificate from a Certificate Authority, which is outside the purview of PKDN.

3. (Receiver) The receiver of a message that has the root-public key corresponding to the Certificate Authority of the sender.

4. (Validator) It is a logically in-line element between the sender and the receiver. It must have the root-public key corresponding to the Certificate Authority of the sender. It verifies the validity of the sender’s certificate and that the certificate has not been revoked. It also sends revocation periodic updates for the sender’s certificate.

Since PKDN does not prescribe any interactions for or with the sender's Certificate Authority, it is not listed as an architectural element. But, the sender, receiver, and validator are expected to be in possession of the root, self-signed certificate of the sender's certification chain-of-trust.

3.2. Root Key Configuration

Every element of the PKDN architecture must be in possession of the root, self-signed certificate of the RM's certification chain-of-trust. Every element of the PKDN architecture, except the RM, must be in possession of the root, self-signed certificate of the sender's Certificate Authority's chain of trust. The root, self-signed certificates must be physically configured in a secure manner on every architectural element. Therefore, the root, self-signed certificates are not expected to change or be revoked.

3.3. Distributed Relationship Management

A relationship in PKDN is defined by the tuple: ((sender-certificate-fingerprint, sender identity, validator identity, receiver identity), E), where:

1. sender-certificate-fingerprint is a cryptographic hash of the sender's public key certificate, which is specified to expire at time CE -- specified in Universal Time (UT);

2. validator identity designates the validator that created this relationship;

3. receiver identity designates the receiver for which the validator created this relationship; and,

4. E is a future time, which is specified in Universal Time (UT), when this relationship must expire such that E is less than or equal to CE.

The relationship is stored asynchronously by the sender, validator, and receiver. Validator stores the relationship tuple first. The receiver and sender store the relationship tuple asynchronously after receiving a communication from the validator.

Let L be a system-wide constant to indicate the maximum duration for the validity of a given relationship. Let M be the maximum expected communication delay in the DTN, over which the PKDN is an overlay. Then, L >> 3*M: the duration of relationship validity must be much greater than three times the maximum expected communication delay anywhere in the DTN. The following expression is a corollary: L/3 >>

M. When L is 8 hours, for example, the communication delay, M, must be less than 2.66 hours. The expiry time for the relationship, E, is computed as E = (T + L) where T is the time when the relationship was created by the validator. The sender, receiver, and validator must asynchronously delete expired relationship tuples. If the validator receives a revocation notice including a sender-certificate-fingerprint, which has unexpired relationships, then the validator must send a revocation notice for that relationship to respective receivers. The sender can prevent the expiry of a relationship tuple by sending a fresh relationship request to the corresponding validator.

Optionally, the sender may have multiple unexpired relationship tuples with a receiver by sending relationship requests through multiple validators. The receiver can reject relationships by sending an unsubscribe message for a specified sender-certificate-fingerprint to the validator.

3.4. PKDN Data Structures

Relationship are created, revoked, or rejected by asynchronously passing messages -- we only assume synchronization of clocks in the PKDN, which is also the assumption in the underlying DTN. The messages passed along with their formats are as follows.

1. (Relationship request bundle or RRqBundle) is sent by the sender to the validator. It contains the sender identity, sender timestamp, sender’s public-key certificate, validator identity, receiver identity, and a signature for the RRqBundle using the sender’s private key corresponding to the sender’s public-key certificate.

2. (Relationship creation bundle or RCBundle) is sent by the validator to the receiver. It contains the RRqBundle that triggered the creation of this bundle along with the expiry time of this relationship (E), validator’s public-key certificate, and a signature for the RCBundle using the validator’s private key corresponding to the validator’s public-key certificate.

3. (Relationship creation acknowledgement bundle or RCaBundle) is sent by the validator to the sender. It contains sender identity, validator identity, receiver identity, sender time stamp, sender-certificate-fingerprint, E, validator’s public-key certificate, and a signature on the RCaBundle using the validator’s private key corresponding to the validator’s public-key certificate.

4. (Relationship revocation bundle or RRvBundle) is sent by the validator to the receiver. It contains the sender-certificate-fingerprint, validator identity, receiver identity, revocation time stamp, and a signature on the RRvBundle using the validator's private key.

5. (Relationship rejection bundle or RRjBundle) is sent by the receiver to the validator. It contains the sender-certificate-fingerprint, validator identity, receiver identity, receiver's public-key certificate, rejection time stamp, and a signature for the RRvBundle using the receiver's private key corresponding to the receiver's public-key certificate.

6. (Relationship termination notice bundle or RtnBundle) is sent by the validator to the sender. It contains the sender-certificate-fingerprint, validator identity, receiver identity, sender identity, termination time stamp, termination reason (revocation or rejection), and a signature for the RtnBundle using the validator's private key corresponding.

The message formats can be serialized using JSON, CBOR, or any other serialization format that is compatible with the DTN Bundle Protocol.

3.5. Relationship Service Design

The relationship services of PKDN to its clients are as follows.

1. (Relationship creation) When a Validator receives a RRqBundle from a sender for a receiver, it:

    1. verifies the authenticity and validity of sender's certificate in the RRqBundle;

    2. verifies the sender's authentication in the RRqBundle;

    3. registers a relationship for the RRqBundle with expiry time E, as explained in the previous section;

    4. constructs and sends a RCBundle to the receiver designated in RRqBundle; and,

    5. constructs and returns a RCaBundle to the sender of the RRqBundle.

2. (Relationship revocation) When a validator receives a revocation notice for a sender-certificate-fingerprint from the CRM, it must construct and send a RRjBundle to all receivers who have a relationship with that sender-certificate-fingerprint. The

validator must construct and send RtnBundle to the corresponding sender with revocation as the termination reason.

3. (Relationship rejection) The receiver may can unsubscribe its interest in a sender-certificate-fingerprint by sending a RRjBundle to the corresponding validator. The validator, in response, must send a corresponding RtnBundle to the corresponding sender with rejection as the termination reason.

The specification of L units of time in the design implies that the sender must send at least one PKDN Bundle after every L units of time in order to keep its relationship with the receiver. In response to the relationship initiation from the sender, the receiver can initiate a relationship with the sender by switching their sender-receiver roles. Thus, PKDN can support simplex and duplex security relationships.

3.5.1. Distribution of CRL

The CRM maintains the master copy of the Certificate Revocation List (CRL) in the system. When a new entry is added to the CRL, the CRM sends the addition as authenticated delta CRL update messages to all registered validators. Upon receiving the authenticated delta CRL messages, the validators must update their local copies of CRL. The local copies of the CRL are then used by the validators to provide relationship revocation service to the clients.

The Detla CRL messages from the CRM has the following structure with two structures: (Delta := list((sender-certificate-fingerprint,CE, RTS)), Auth := CRM-authenticator), where Delta is a list that has been added to the master CRL by the CRM and Auth is the digital signature on Delta by the CRM. The list elements of Delta are 3-tuples such that sender-certificate-fingerprint and CE (certificate expiry) are as described in Section 3.3 and RTS is the timestamp when this tuple was added to the master CRL. As with other data-structures, the message format can be serialized using JSON, CBOR, or any other serialization format that is compatible with DTN.

3.6. Reliability and Availability

The reliability and availability aspects of PKDN design are discussed below. The degree of reliability and availability are dependent on the domain of application of DTN and PKDN. Therefore, generic discussions are provided in this section for developing DTN and PKDN with suitable degrees of reliability and availability.

3.6.1. Reliability against misconfiguration

Every client must be configured with the network identifier of its validator. This configuration is not a system wide constant. This information may be configured statically or dynamically using discovery protocols or remote administration protocols before the sender/receiver can join PKDN. It is essential that the configured validator services are reliable and reachable.

3.6.2. Availability

PKDN has two types of network services, namely those for: relationship management and distributed CRL management. The availability of these two network services despite adversarial presence determines the availability of PKDN. As is the case with DTN, PKDN will have to rely on the lower layers to provide availability guarantees despite adversarial interactions.

4. IANA Considerations

This document potentially contains IANA considerations depending on the design choices adopted for future work. But, in its present form, there are no immediate IANA considerations.

5. Security Considerations

Security issues and considerations are discussed through out this document.

6. References

6.1. Normative References

[I-D.ietf-dtn-bpbis]
Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol", draft-ietf-dtn-bpbis-03 (work in progress), March 2016.

[I-D.ietf-dtn-bpsec]
Birrane, E., Pierce-Mayer, J., and D. Iannicca, "Bundle Protocol Security Specification", draft-ietf-dtn-bpsec-00 (work in progress), December 2015.

[I-D.templin-dtnskmreq]
Templin, F. and S. Burleigh, "DTN Security Key Management - Requirements and Design", draft-templin-dtnskmreq-00 (work in progress), February 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <http://www.rfc-editor.org/info/rfc4838>.

[RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, DOI 10.17487/RFC5050, November 2007, <http://www.rfc-editor.org/info/rfc5050>.

[RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, DOI 10.17487/RFC6257, May 2011, <http://www.rfc-editor.org/info/rfc6257>.

6.2. Informative References

[CAP] Brewer, E., "CAP twelve years later: How the "rules" have changed", Feb 2012, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6133253>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <http://www.rfc-editor.org/info/rfc5751>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://www.rfc-editor.org/info/rfc6066>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <http://www.rfc-editor.org/info/rfc6960>.

Authors' Addresses

Kapali Viswanathan
Boeing Research & Technology
Unit 501, 5th Floor, Tower D, RMZ Infinity
No 3, Old Madras Rd
Bangalore, KA 560016
IN

Email: kapaleeswaran.viswanathan@boeing.com

Fred L. Templin
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org