NDN, Technical Report NDN-0054. http://named-data.net/techreports.html
Revision 1: March 19, 2017

NDN Certificate Bundle (Version 0.1)

Manika Mittal, Alex Afanasyev, Lixia Zhang

Abstract—**The Certificate Bundle is designed for use in systems that use the Named Data Networking (NDN) architecture. The Certificate Bundle protocol provides a way to retrieve a set of the certificates needed to authenticate a data packet within one RTT and assures that the certificates for the authentication can be retrieved from the same place as data packet (original data publisher or a managed repo).**

1 Introduction

Over the years, the use of computer networks has seen a major shift: from resource sharing to content distribution, from caring about "where" the resource is to caring about "what" the content is. Named Data Networking (NDN) transforms the current host-centric network architecture of IP to a data-centric network architecture. Instead of delivering the requested data packet to the desired destination address, the data consumer can fetch the desired data by specifying its unique name. At a very high level, the consumer will send out an Interest packet specifying the name of the desired data, and the network is responsible for finding the corresponding Data packet and returning it to the consumer.

NDN architecture brings several advantages that pose considerable issues in the current IP architecture. While some of these advantages include making the content more available and location independent, one major feature of NDN is its built-in data authentication mechanism. NDN builds data authentication into the network layer by requiring that all the Data packets are signed.

Every NDN Data packet *[2]* consists of the following fields: Name, MetaInfo, Content and Signature. The Signature field is defined as two consecutive TLV blocks: SignatureInfo and SignatureValue. The SignatureInfo block describes the signature and consists of fields like SignatureType and KeyLocator. The KeyLocator field specifies the Name (or KeyDigest) that points to another Data packet containing the certificate or the public key that is needed to validate the Signature Value. Since the certificate is itself an NDN data packet, it has a Signature field of its own.

The relationship between the data names and the key names, which are used to validate the data signature, depends on the application's trust model. Trust schema *[1]* is a set of linked trust rules and trust anchors that concisely describes the trust model of an application and is used to determine the authentication paths.

In order to authenticate a Data packet, it is first checked against the trust schema (e.g., by an automated schema interpreter). If the name of the Data and the name of the key signed the data satisfy trust schema rules, one needs to use key name to fetch the Data packet of the key and check it recursively against the trust schema. When a trust anchor (a key whose trust is pre-configured by some means) is reached, one can follow the chain and verify correctness of each signature.

The above procedure highlights the need to obtain a chain of certificates to verify a signature poses several challenges, including (1) the need for multiple non-parallelizable data retrievals and (2) the need to ensure availability of certificates.

The Certificate Bundle protocol is designed to solve the above mentioned issues with signature verification in NDN. The idea is to allow data publisher to also publish all the certificates that are needed to authenticate the published Data packet as a "bundle". This way, instead of retrieving every single certificate of the chain one by one, one can retrieve the whole chain as a bundle Data packet(s). Moreover, since all the certificates are kept together, it improves availability of certificates as the bundle can be published/provisioned together with the data packets they authenticate.

2 Overview

In order to understand the working of certificate bundle let's consider the data consumer and the data producer side separately.

2.1 Consumer Side

At the data consumer side, the goal is to validate the signature of a particular data [target data packet]. In order to do that the consumer needs to fetch the certificate (to validate the signature) as well as all the certificates needed to validate the first certificate. With the help of a certificate bundle, the consumer can retrieve all the certificates with a single interest. More than one interest might be needed in case the bundle has many segments and each segment can be retrieved as needed (e.g., the second segment might not be needed when the consumer previously validated a similar data packet).

The retrieved certificate bundle is not guaranteed to contain all the certificate needed for validation. Whenever a necessary certificate is missing from the bundle, the client should try to fetch the remaining certificates directly.

2.2 Producer Side

The goal of the bundle producer is to create the certificate bundle to the best of it's knowledge. It's important to note that the goal of a certificate bundle is merely to help with the validation process by providing a collection of certificates that are may needed for the validation of a data packet. It is possible that not all of the certificates in the bundle are used for validation. For example the bundle producer might use the following procedure to create a bundle for single hierarchy type trust models.

To create a bundle, the key locator field or the signing key name of the data packet [for which the bundle is being created] is needed. From this point on, the producer fetches all the certificates required to validate the key till it reaches a "dead-end". The definition of "dead-end" is given below. On the producer side, the concept of trust-anchor is undefined. Currently we don't assume that the producer knows the trust anchor of the verifier as it's possible under some trust model that there is no fixed trust anchor.

The producer includes all the certificates in the bundle till it reaches a "dead-end". The producer reaches the "dead-end" if any of the following conditions are satisfied:

1) If the producer reaches a self-signed certificate. This certificate in currently included in the bundle. This may change in future.
2) If the producer reaches a loop i.e it encounters a certificate that has already been included in the bundle. This certificate is NOT added to the bundle again.
3) If the producer reaches a maximum limit on the number of certificates. This limit is currently set to 25 certificates.
4) If the producer reaches a malformed certificate. This is NOT included in the bundle.
5) If the producer reaches a certificate that either doesn't have a Key locator field or has a key locator field of a different type. This situation is probably rare. Currently such a certificate is included in the bundle. This may change in future.

Note it's possible a bundle is created and published even if the producer has not retrieved ALL the certificates till the "dead-end". In such a case, the later versions of bundle will include the complete chain. It is the verifier's responsibility to fetch the latest version of the bundle or the remaining certificates individually.

At any point the producer may refresh the current bundle state which effectively fetches all the certificates again in order to ensure the latest versions of all are included in the bundle.

3 Functional Specification

3.1 Naming Conventions

The name of the Certificate Bundle is an extension of the name of the target data packet which is to be verified. It starts with the associated name followed by a special name component _BUNDLE, indicating that the content is a Certificate Bundle. After the _BUNDLE component, there is a version number because it's possible that the Certificate Bundle is updated (in case some key in the chain is revoked). The last component of the Certificate Bundle name is a segment number in case the Certificate Bundle is too big to fit in one data packet. Note that the segment number 0 is present in the name of the first segment of Certificate Bundle even if there is only one segment of the bundle.

Hence the naming convention for the Certificate Bundle is as follows:

```
/<derived(data_name)>/_BUNDLE/<trust-model>/
↪<version>/<seg>
```

where the derived(data_name) is determined by the naming rules.

The current specification defines the following rule:

1) If last name component is segment number, then:

```
derived(data_name) = data_name.getPrefix(-1)
```

Other rules will be defined in later versions of this specification.

In the implementation, currently, the <trust-model> component is the number 00 which signifies single hierarchy. The current bundle can help with the certificate retrieval for any trust schema that uses a single chain of certificates to verify a data packet (a.k.a single hierarchy).

3.2 Certificate Bundle Packet Format

The Certificate Bundle is an encapsulation that consists of a list of certificates needed to authenticate an NDN Data packet.

A certificate bundle may have many segments if the list of certificates exceed the maximum packet size. Each segment must always have complete certificates.

A Certificate Bundle packet consists of multiple certificates starting with the certificate needed to validate the original data packet, followed by the certificate needed to validate the previous certificate, and so on. The certificates in the certificate bundle should appear in the order in which they are needed for validation. The certificates in the bundle are kept in a sorted order to avoid unnecessary fetching of bundle segments. The bundle segments are fetched only if more certificates are needed to validate a data packet, so if the certificates are kept in a random order this might lead to additional fetches which can be avoid if the certificates are in sorted order.

The Certificate Bundle packet uses DigestSha256 signature which provides no provenance of the packet and is intended to protect against any unexpected modifications.

The Certificate Bundle Data packet is a TLV defined as follows:

```
CertificateBundle ::= DATA-TYPE TLV-LENGTH
        Name
        MetaInfo
        BundleContent
        Signature

BundleContent ::= CONTENT-TYPE TLV-LENGTH
                  Certificate+
```

The list of certificates in the BundleContent SHOULD be in the following order - the certificate to validate the target data packet, followed by the certificate to validate the first certificate and so on.

4 Known Limitations

1) The current limitation of the certificate bundle is the lack of ensuring that the verifier has the latest version of the certificate bundle. In case a certificate present in the bundle expires or is revoked, ideally we would like to supply the verifier with the latest bundle with updated certificates. The current implementation uses parameters like freshness period and must be fresh, which merely mitigates and the problem and does NOT guarantee that the verifier gets the latest version of the bundle.
2) Currently, bundle fetching is only attempted for data validation.

5 Acknowledgement

This work is partially supported by the National Science Foundation under awards CNS-1345318 and CNS-1629922.

References

[1] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. Schematizing trust in Named Data Networking. In *Proceedings of 2nd ACM Conference on Information-Centric Networking*. September 2015. URL: http://dx.doi.org/10.1145/2810156.2810170.

[2] NDN Project Team. NDN packet format specification (version 0.2-2). http://named-data.net/doc/ndn-tlv/, 2017.