Computer Networks 112 (2017) 1–11

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Up-to-date key retrieval for information centric networking[☆]

Giulia Mauri[*], Giacomo Verticale

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, Milano, Italy

ARTICLE INFO

Article history:
Received 28 November 2015
Revised 23 June 2016
Accepted 31 October 2016
Available online 1 November 2016

Keywords:
Information centric networking
Named data networking
Digital signature
Public key updating
Key revocation
CRL
OCSP

ABSTRACT

Information Centric Networking (ICN) leverages in-network caching to provide efficient data distribution and better performance by replicating contents in multiple nodes to bring content nearer the users. Since contents are stored and replicated into node caches, the content validity must be assured end-to-end. Each content object carries a digital signature to provide a proof of its integrity, authenticity, and provenance. However, the use of digital signatures requires a key management infrastructure to manage the key life cycle. To perform a proper signature verification, a node needs to know whether the signing key is valid or it has been revoked. This paper discusses how to retrieve up-to-date signing keys in the ICN scenario. In the usual public key infrastructure, the Certificate Revocation Lists (CRL) or the Online Certificate Status Protocol (OCSP) enable applications to obtain the revocation status of a certificate. However, the push-based distribution of Certificate Revocation Lists and the request/response paradigm of Online Certificate Status Protocol should be fit in the mechanism of named-data. We consider three possible approaches to distribute up-to-date keys in a similar way to the current CRL and OCSP. Then, we suggest a fourth protocol leveraging a set of distributed notaries, which naturally fits the ICN scenario. Finally, we evaluate the number and size of exchanged messages of each solution, and then we compare the methods considering the perceived latency by the end nodes and the throughput on the network links.

1. Introduction

Named Data Networking is funded by the National Science Foundation for the Future Internet Architecture project. Projects such as Named Data Networking (NDN) and Content Centric Networking (CCN) belong to the same program for defining a network where the focus is on “what” users care about and not on “where” they are. In this novel architecture, generally called Information Centric Networking (ICN), contents are addressed by their name and not by their location. Thus, the attention is shifted from users to content, resulting in a caching network that is more efficient and flexible than an IP network for content distribution and management with beneficial effects on timely delivery. Moreover, the validity of a content depends on the validity of the signature on the data packet, differently from IP network where data security depends on the transmission channel. Such content centric architecture rises up new security challenges related to key management that should be addressed.

In NDN, the content objects are divided into chunks, each digitally signed by its producer. Otherwise, the content chunks are organized together with their digest into a Manifest, which is signed by the producer. Thus, the public key management becomes a crucial issue for ICN security. Even if each node could verify the signature before caching objects, most papers assume that verification is made only by the content consumer. Indeed, in order to perform the signature verification, a node needs the signer’s public key, which can be easily retrieved by issuing a standard interest message. However, information about the key validity status is also necessary. In fact, a content signed with a compromised key may remain in cache for an indeterminate amount of time, and possibly be served to the end users. Even if caches implement a freshness mechanism that deletes a content that has been in the cache longer than a given threshold, a compromised node could resend data making extremely difficult to remove from the network the objects signed with a compromised key, resulting in a denial of service and paving the way for more sophisticated attacks.

The data object authentication is one of the research challenges presented in the IETF draft [1]. The problem is also analyzed in a survey of security attacks in ICN [2]. Indeed, there is an urgent need to define and support a mechanism to distribute updated publisher’s public keys to the consumers of data objects. In the standard PKIX (Public Key Infrastructure Certificate X.509), the issue of delivering key revocation status to the end nodes is

[☆] A preliminary version of this paper appears in G. Mauri, and G. Verticale, “Distributing key revocation status in named data networking”, in 19th EUNICE Workshop on Advances in Communication Networking, August 2013.

[*] Corresponding author.
E-mail addresses: giulia.mauri@polimi.it (G. Mauri), giacomo.verticale@polimi.it (G. Verticale).

http://dx.doi.org/10.1016/j.comnet.2016.10.018
1389-1286/© 2016 Elsevier B.V. All rights reserved.

solved by using the CRL (Certificate Revocation Lists) [3] or the OCSP (Online Certificate Status Protocol) protocol [4]. The current approaches in PKI for key management and revocation are far from optimal and the research community is actively looking for new techniques. Nonetheless, no current ICN proposal provides a key revocation component, which is necessary for wide-scale deployment. Therefore, we are the first considering the most obvious solution, i.e. porting PKI solutions to the name-based environment. We also assess their performance in the NDN framework. Similarly, we also add a novel approach that preserves the distributed nature of the ICN networks. We compare it to the other approaches and clarify what are the advantages and shortcomings of such protocol. We present a solution based on the ccnx-repository synchronization protocol that implements similar functionalities to CRL. So far, this is the only solution for key management that is included in a protocol specification for the ICN scenario. Then, we suggest two reactive methods that recall the principles behind the OCSP for the Information Centric Networking framework. In particular, the nonce-based scheme always retrieves the original key from the producer, and the timestamp-based method exploits timestamps over the keys to guarantee freshness.

Finally, we consider a notary-based method like Perspectives [5] that is an alternative to the traditional PKIX for authenticating the public keys. Indeed, we propose how to get up-to-date keys retrieving them from the nearest nodes in an NDN-friendly way.

The main contributions of this paper are:

- We provide, as far as we know, the first proposal to adapt the OCSP and CRL schemes to the ICN scenario.
- We propose a new solution based on the concept of notaries, to better adapt the up-to-date key retrieval in a NDN-friendly way.
- We evaluate and compare the various proposals in terms of number of exchanged messages, latency, and throughput.
- We show that our solution overcomes the main drawbacks of the standard schemes guaranteeing good network performance. However, it is not always superior to the other protocols, e.g., when scalability and resistance to network partitioning requirements can be relaxed, a timestamp-based reactive protocol is better.

The remainder of the paper is structured as follows: Section 2 reviews the related work and Section 3 recalls some background notions, together with the description of the communication protocol in Section 3.3. Section 4 presents the network scenario and the attacker model together with the security definition. Section 5 proposes the traditional methods to distribute valid key and shows our novel scheme. Finally, Section 6 describes the evaluation scenario, and gives the performance results before the conclusions that are left for the last Section 7.

2. Related work

The public key management is a main security issue in a content centric network. Since the first work on CCN, Smetters and Jacobson [6] rise the problem of content authentication. Their proposal is to authenticate the link between packet name and content using a digital signature scheme implemented by the content producer. In this way, they provide guarantee of content validity and origin. However, they do not inspect technical problems related to the signature scheme used, for example, how to check the signing key status, i.e. if it is compromised or revoked. The challenge for the definition of a public key management in NDN is presented and evaluated in [7]. The certificate format, distribution and revocation are discussed providing a starting point for the definition of a PKI in NDN. The signature and key revocation are presented from the point of view of the data producer/private key owner. This is a good starting point for our work, which on the other hand analyzes the same problems from the point of view of the data consumer/public key owner.

A recent work [8] suggests a trust management scheme for NDN. The paper provides the definition of the Interest-Key Binding (IKB) rule, that is also exploited along our paper. The IKB rule binds the producer's public key with the consumer's interest. Our last two assumption presented in Section 4.1 are complementary to the IKB rule. However, our paper provides a solution for the management of key revocation that is left open in [8].

The paper [9] suggests a Key Resolution Service (KRS) for CCN, that is, as far as we know, one of the few proposals relative to key management in ICN. This service allows to map a content name with the corresponding security information. The KRS is queried by the consumer node before sending the interest for a content. Thus, the consumer can obtain the public key certificate of a publisher or the content digest. This solution is a first practical attempt to mitigate content poisoning attack. The main drawback of the proposal is the presence of a local KRS server that could become a bottleneck. Moreover, the same paper presents some performance results relative to the average latency per request sent to the KRS server. The latency is measured as a function of the cache size and the number of KRS servers. Our work differs from [9] because we do not need a new network entity such as the KRS, we allow the consumer to choose its security window and also the keys are frequently refreshed. The authors of [10] present a platform used to obtain performance results about CRL and OCSP. The authors show the temporal behavior of CRL and OCSP in terms of the processing time. The results relative to CRL are shown to be around 1 ms and those relative to OCSP are between 25 and 30 ms. These results are obtained over a standard IP network and end up in a small delay over the performance gathered with no certificate management. Our paper extends those protocols for ICN showing that the resulting latency is comparable to the ICN setting with no key revocation management.

The paper [11] suggests a distributed architecture which exploits cothorities to validate and sign certificates, timestamps and log records. The work proves that using collective authorities is practical and also provides higher link security than today's centralized authorities. The proposal is similar to our proposed notary-based protocol where some trusted nodes perform the certificate checking. However, our solution is implemented for the Information Centric Networking scenario.

Thus, if a public key management infrastructure is not defined, there can happen that invalid or revoked public keys are spread and used into the network. Therefore, a Denial of Service attack is easily exploitable. The problem is inspected by various papers and here we report some of them.

Gasti et al. [12] present a first attempt to identify and mitigate DoS and DDoS in ICN. Particularly, the paper describes two types of attacks: interest flooding and content/cache poisoning. The first threat consists on sending a large number of interests requesting contents from the same set of producers; the second aims to cause node to cache corrupted or false content objects, obstructing the retrieval of legitimate contents. The paper also discusses tentative countermeasures against the attacks but it does not evaluate their efficacy with a simulation. The paper represents a first step into the definition of content/cache poisoning in NDN scenario, but it does not consider the problem of key validity.

The papers [13] and [14] analyze the interest flooding-based DDoS over NDN, considering that interests are sent for non-existent contents and obviously fill up the victim's PIT. The authors in [13] propose proactive and reactive countermeasures, but then, they focus on reactive methods for detection of interest flooding via junk interests. Moreover, the authors propose a mitigation technique, called Poseidon, which identifies traffic anomalies and

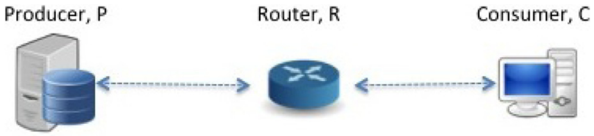

Fig. 1. The reference scenario.

keeps several statistics on expired interests. The authors in [14] define three mitigation methods with varying degree of implementation complexity against the attack. Particularly, they propose a token bucket approach and two satisfaction-based methods. Finally, the papers evaluate the benefits of the countermeasures through small-scale and large-scale simulations over real network topologies. However, the proposals cannot be applied to the problem of content pollution because they are based on interest packets and also they do not consider the problem of retrieval of key status.

The content poisoning attack in NDN is studied in [15]. The authors suggest a content ranking algorithm for cached content to allow routers to distinguish between a fake or a valid content. When the consumer verifies the signature and detects a fake content, it sends an interest that excludes the received content. Thus, the router can assign a rank to each cached object that is updated when an interest for that content is received. The content with an highest rank is selected as response to an interest packet. The proposal is shown to be effective against content poisoning. However, the problem of key revocation still remains open.

Thus, there is an urgent need for the definition of a scheme that opens the possibility to retrieve updated keys in the ICN scenario exploiting and adapting the well known solution for IP-based networks. Indeed, as also stated in [16], none of the previous work have addressed the issues relative to security aspects about key management. This paper inspects the problem of content poisoning and suggests three PKI-like solutions for the key management and, then, presents a novel method that reflects the NDN distributed nature.

3. Background

This section gives some preliminary concepts about the information centric architecture and protocol.

The ICN architecture comprises three different nodes, as shown in Fig. 1:

- **Data Producer, *P***: upon reception of an Interest packet, it answers with the corresponding Data packet. It signs a content by using its key.
- **Data Router, *R***: upon reception of an Interest packet, it answers with the corresponding Data packet, if it is present in its content store. Otherwise it forwards the request towards the correct Data Producer. Upon reception of a Data packet, it forwards it to the downstream Consumer. Moreover, it caches packet in its Content Store.
- **Data Consumer, *C***: obtains data sending Interests with the desired data name.

Moreover, the architecture comprises a Trusted Authority (TA) that periodically updates the public/private key pairs. In the remainder of the paper, we assume that the communication network is reliable and timely, i.e., no message can be lost due to communication delays or node malfunctioning.

3.1. Signature generation and verification

The content producer, P, is responsible for the digital signature over the content, C, and the corresponding name, N. Particularly, a content is made available in the network as $M_{N,C,P} = (N, C, Sign_P(N, C))$, where $Sign_P(N, C)$ is the producer's signature over the name and the content. The signature generation can follow one of the two forms: single blocks are individually signed using a standard public key algorithm, e.g. RSA with SHA256, or multiple blocks are signed together with an aggregated signature scheme, e.g. Merkle Hash Trees [17]. A content consumer retrieves the content, C, using its name, N and it should be able to find the public key to use to verify $Sign_P(N, C)$. How the consumer finds the key is explained in the next subsection.

3.2. Key model

In the following sections, we follow the key trust model presented in [18] and [19]. As depicted in Fig. 2, a root key signs the site keys, which in turn sign the user's keys. Then, each user is responsible to sign and to maintain in a local repository the device and application keys. This model allows users to follow the trust chain from the leaf nodes (i.e. application and device) to the root key for verifying the validity of a key. In each Data packet, there is a "KeyLocator/KeyName" field, that one can use to fetch the key. Moreover, since the key itself is a Data packet, the key "KeyLocator/KeyName" field is used to reach a trusted anchor. The root key, that is "public knowledge" and self-signed, is used to verify the key needed to verify the Data packet. Furthermore, the Fig. 2 shows the key name structure. Usually there is a common prefix "/keys", used to easily distinguish keys from contents, as the first part of the name; the middle part represents the path in the keys subtree, namely the keys hierarchy in the network; the final part is the hash value of the corresponding public key. There could be another part of the name that carries the content version and segment, but it is not mandatory.

3.3. Simplified model of the ICN protocol

The basic information centric protocol follows the request/response paradigm. A response is not given back if it is not received a request. The request message is called Interest, whereas the response Data Packet. Each Interest is uniquely identified by a Name and a Nonce. Then, the corresponding Data Packet must carry the same Name and the same Nonce. The Data Packet is always signed by its Producer following the RSA signature algorithm. From now on, an Interest packet is represented by $\mathtt{I(name)}$ and a Data packet by $\mathtt{D}_{signer}\mathtt{(name)}$.

The simplified communication protocol consists of 7 phases:

1. **Setup**: the initial phase is performed only once to define the set of public parameters and to distribute them to the users.
2. **Key Gen**: this phase is performed time to time to generate the key pairs and to distribute them to the users. The Producer runs the key generation algorithm and gets (pk, sk). Then, it pushes to the TA the public key to be certified and keeps the private key secret. Finally, the Producer publishes off-line the public key pk into the nodes' repositories by means of the ccnx synchronization protocol.
3. **Create Data**: the Producer, P, produces and stores contents. Each content is represented by $m = \mathtt{D}_P\mathtt{(name\_m)}$ and the corresponding signature is $\sigma \leftarrow Sign_{sk_P}(H^s(m))$.
4. **Send Interest**: the Consumer, C, sends an Interest $\mathtt{I(name\_m)}$ with the name of the data it wants to the next hops.

5a) **Forward Interest**: if the next hop, a Router R, does not have the content, it forwards the Interest $\mathtt{I(name\_m)}$ to the next hop.

5b) **Send Data**: if the next hop, a Router R or the Producer P, has the content, it answers with the corresponding data $\mathtt{D}_P\mathtt{(name\_m)}$.

6) **Receive Data**: the Consumer, C, receives the data $\mathtt{D}_P\mathtt{(name\_m)}$.

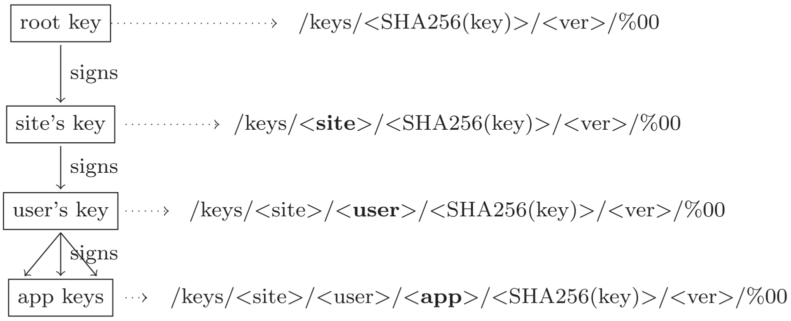
root key
/keys/<SHA256(key)>/<ver>/%00
signs
site's key
/keys/<**site**>/<SHA256(key)>/<ver>/%00
signs
user's key
/keys/<site>/<**user**>/<SHA256(key)>/<ver>/%00
signs
app keys
/keys/<site>/<user>/<**app**>/<SHA256(key)>/<ver>/%00

Fig. 2. Key model and naming in NDN.

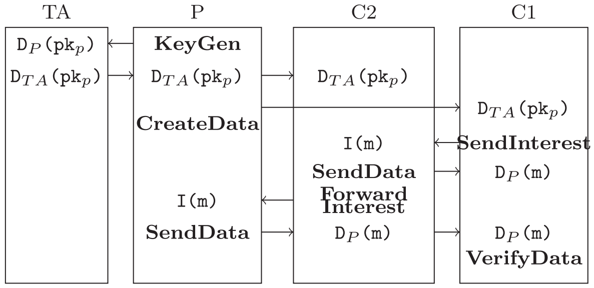

Fig. 3. The communication protocol.

7) **Verify Data**: the Consumer, C, verifies the content: $Ver_{pk_p}(H^s(m), \sigma') \overset{?}{=} 1$.

The following Fig. 3 shows the message exchange.

We suggest to verify all the packets at the end nodes to prevent cache pollution attacks or more sophisticated attacks. The standard verification procedure follows the protocol. However, the signature validation is a crucial problem. Each node needs to check the public key status, as described in the following.

The verification process should follow the next steps. After receiving a Data Packet, each end node in the network:

1. Checks the `KeyLocator` containing the signing key name or the key itself.
2. Checks if the key name is known and if the key is stored in the cache. If a match is found, the node checks the key status. If the key status was checked in the security window, then the Data packet can be verified.
3. Otherwise, the node expresses an Interest for the key, and waits till the reception and the validation of the key before verifying and accepting the content into its cache.

3.4. CCNx synchronization protocol

Every CCN node has a repository where content objects are persistently stored in addition to the content store. The CCNx synchronization protocol allows repositories to be automatically up to date [20]. A set of contents whose prefix name is common, is called Collection. The contents in the collection are organized within a sync tree, that is built by the local Sync Agent. The agent computes an additive hash over that tree, the topmost hash is called root hash. Periodically, the agent sends Root Advise Interests to the neighboring nodes which synchronize their repositories comparing the root hash of the sync tree sent in the Interest with their own root hash. If they match, the collections are in synchronization. If they do not match, the collections are updated using the standard interest/data protocol and sending the different root hash between the nodes.

4. The up-to-date key retrieval security problem

4.1. Assumptions

1. Each Data packet contains a `KeyLocator` field containing the name of a public key, which can be fetched with the standard ICN mechanism.
2. A public key is valid if it is included in a certificate issued by a Trusted Authority and if there is a proof that it has not been revoked. A vulnerability period of duration W is acceptable.
3. The owner of a valid key is honest and only signs the contents that it is authorized to sign. This paper does not discuss how to scope signatures or enforce name-key binding rules.
4. All the contents signed with an invalid key must be dropped by the Consumer, even if they were signed when the key was valid. This paper does not discuss how to remove stale content from the router caches.
5. We assume that each node has a clock that increases at the same rate of the others with a tolerance of ± $0.05s$. However, the model is partially synchronous meaning that the clocks may display different values at the same time.
6. The message delivery time t_{d_B} from node A to node B and the message processing time t_{p_B} at node B can be measured and are known for each node. If no Data packet is received within $RTT = 2 \cdot t_{d_{TA}} + t_{p_{TA}}$ after the Interest has been sent, then the node deduces that the message is lost or no correspondent Data exists. RTT is called the Round Trip Time.

4.2. Attack scenario and security definition

Our attacker model assumes an active, dishonest node, which can inject any content into the node caches. In particular, our attacker:

1. can obtain any valid content produced and signed in the network.
2. can insert any content of its choice in any Router content store.
3. can obtain any Producers' private key. In this case, the Producer immediately revokes the stolen key.
4. cannot obtain more than a single key in an interval W.
5. cannot break any cryptographic algorithm.

According to the attacker previously described, we provide our security definition:

Definition 1. The key retrieval scheme is secure if no uncompromised Consumer accepts as valid a content signed with an invalid key, except for the case that less than a time W has passed since the key was revoked.

5. Key retrieval schemes

ThisSection reviews and provides additional details about the three key retrieval schemes that have been firstly presented in [21]. Then, we describe the proposal of this paper which overcomes the drawbacks of the three previous schemes.

The first protocol supposes to create a list with valid keys that are updated and signed by the Trusted Authority (TA), and then distributed to the network nodes using the ccnx synchronization protocol.

Moreover, we recall the two reactive protocols: the nonce-based and the timestamp-based schemes. Indeed, a user sends an Interest for a key to the Producer, and the latter answers with a Data packet containing the key and its signature. The main difference is the validity window that it is very small for the nonce based protocol and a chosen value from the users in the timestamp based protocol.

The fourth protocol exploits some trusted nodes that are responsible for providing a specific key, when they receive an Interest addressed to themselves. This solution easily fits the distributed nature of the Information Centric Networking paradigm.

On the one side, the original ICN communication protocol does not need pervasive modification, the communication follows the standard Interest/Data packet exchange. On the other side, the Interest and Data packets need some changes. In particular, the Consumer sends an Interest for a key specifying the requirements for that key, as detailed in the following. Then, Producer or Routers answer with the corresponding key following the required criteria.

5.1. Protocol 1: proactive method

Protocol P1 periodically distributes up-to-date keys to consumer nodes. Such predistribution can leverage either the proposed CCNx synchronization protocol [20] or, alternatively, the ChronoSync protocol [22].

The Trusted Authority and each Consumer keep a repository holding the public keys. Upon every key update, the TA pushes modifications to all the Consumers.

In order to enable key retrieval, the Trusted Authority defines a *key collection* where the Producers' public keys are listed.

The keys in the collection are then organized within a tree and the TA computes a *root hash* over that tree.

Whenever some change happens, a RootAdvise message with the root hash is sent to all the consumer nodes. The Consumer compares the root hash of its repository with the received root hash. If the hashes are equal, the repositories are up-to-date; otherwise the Consumer expresses an Interest to request the keys that have been modified.

Fig. 4 shows the synchronization process. Notice that after the Consumer has compared the hashes, it sends Interests for the keys that are not up-to-date.

The main advantage of this solution is that the key repositories are kept synchronized and, therefore, no key retrieval is necessary when data arrives. On the other hand, the main disadvantage is that Consumers must keep a large number of keys for Producers even if they are not interested in their content. Additionally, as the number of keys grows, the update messages are more and more frequent resulting in a significant overhead. Additionally, the synchronization procedure must be repeated for each Consumer, potentially violating the security window for some other Consumers.

This key retrieval scheme is secure for whichever security window W because the key repositories are synchronized every time there is a key modification or revocation, except that the synchronization procedure incurs in significant delays.

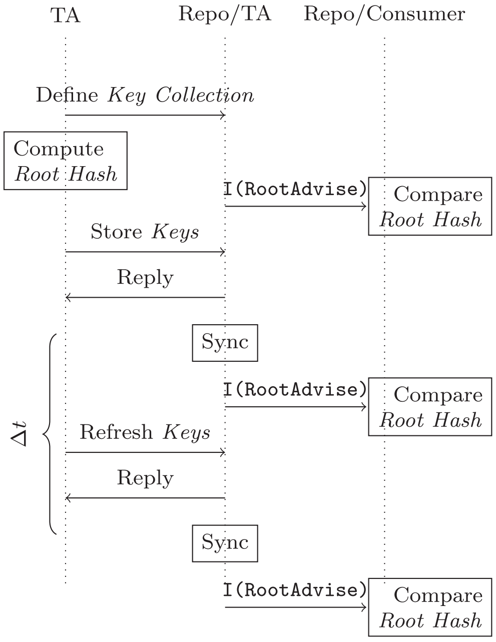

Fig. 4. Repository Synchronization. Note that the RootAdvise message is a special Interest message.

5.2. Protocol 2: nonce-based

Protocol P2 guarantees the up-to-date status of the key ensuring that the key is sent directly by the Trusted Authority.

Whenever a new Data packet arrives, the Consumer checks whether the corresponding key is available and executes Algorithm 1. If no valid key is found, an Interest for the key is sent by the Consumer node. The Interest must contain the "do not answer from content store" option and a nonce. Thus, the TA answers with a Data packet containing the root key and the same nonce.

Algorithm 1 Verification of content validity.

```
Check the KeyLocator.
Check the key in the Content Store.
if a match is found then
    Check the key status.
    if Key status was checked in the window then
        Verify the Data packet.
        return
    end if
end if
Send an Interest for the key.
Wait for the key then check key and content.
```

Particularly, the Consumer node sends an Interest specifying the following fields: name=key/pk_P, selector equal to answer origin kind, meaning "do not answer from content store", and a unique nonce, necessary for guaranteeing the uniqueness of the response. Note that key/pk_P is the name of the Producer's public key.

As soon as the Consumer receives the Data packet containing the key, it verifies that the message is signed by the TA and that the message includes the unique nonce, which is also part of the authenticated data. Then, the Consumer verifies the original message and stores the key in the Content Store. Algorithm 2 describes the operations performed by the Consumer.

Algorithm 2 Consumer in protocol P2.

```
Send I(key/pk_$P$). This message includes a random nonce, $n$.
Wait until D_$P$(key/pk_$P$) is received.
Check that D_$P$(key/pk_$P$) includes $n$.
Check that D_$P$(key/pk_$P$) has been signed by a TA and the signature is valid.
Store the key in the CS.
Use the key to check the signature of pending messages.
```

It is worth noting that, by virtue of the `answer origin kind` option, the Router nodes only forward the Interest packets to the following nodes toward the TA.

The latter answers with a Data packet containing the `name=key/pk_`P, the `signed info` that are the same nonce sent in the Interest packet and the Publisher Public Key Digest that identifies the Producer, and the content that is the public key needed to verify the original packet. These data are signed by the TA.

Then, the Data packet is forwarded from the Trusted Authority to the Consumer.

The following Algorithm 3 describes the operations performed by the TA.

Algorithm 3 Trusted authority in protocol P2.

```
Receive I(key/pk_$P$) with nonce $n$.
Put $n$ in D_$TA$(key/pk_$P$).
Forward D_$TA$(key/pk_$P$) to the Consumer.
```

This key retrieval scheme is secure if the security window respects the following condition: $W > RTT$, i.e. it should be bigger than the round trip time, RTT, between the Consumer and the Trusted Authority. Note that the tightest security window can be achieved only with the exact knowledge of the RTT. However, W can be chosen to be larger than the maximum network RTT. On the one hand, this protocol can guarantee security with very small windows, W, since the key is guaranteed to be fresh after each execution of the protocol. On the other hand, the TA can become a bottleneck.

5.3. Protocol 3: timestamp-based

In Protocol P3, each incoming key is signed by the TA along with a timestamp. When the Consumer needs a key, it checks in the Content Store. If a matching key with timestamp T_0 is found, then the Consumer checks whether $T_0 + W$ falls later than the current time, in which case, the key is considered valid. Otherwise, the Consumer sends an Interest for the key to all its neighbors specifying in the name a timestamp, TS, that indicates a threshold validity.

The Consumer sends an Interest packet with `name=key/pk_`P/TS. A Router, having in its Content Store the key `key/pk_`P/T_r with $T_r > TS$, can answer with the corresponding Data packet.

Otherwise, the node forwards the Interest to the following node up to, possibly, the TA. The TA is assumed to generate key messages on-the-fly with the current time. Notice that system-wide clock synchronization is necessary for the correctness of the protocol.

This key retrieval scheme is secure if the security window is bigger than the chosen timestamp $W > TS$. The Consumer can choose any value for TS which is smaller than the current time and larger than the current time minus W. A small value will result in a quicker response from the network, but also in more frequent key expirations. A larger value will result in a higher latency in obtaining the key, but in longer key durations.

5.4. Protocol 4: notary-based method

Protocol P4 further enhances the scalability of the previous protocols by leveraging on specially trusted nodes, called Notaries, to provide keys on behalf of the TA. These Notaries are chosen by each Consumer and located in diverse network locations near the end users.

When the Consumer needs a valid key, it sends an Interest packet to a set of Notaries of size N_S asking for the content with `name=notary/`N_i`/key/pk_`P, where N_i is the notary's name. This Interest can only be satisfied by the Notary as in Protocol P2. This protocol requires to add a FIB entry for each Notary in all the network nodes, so that they can route packets to *named notaries*. The other part of the name specifies that the content will be a key, `/key`, and in particular, the public key of Producer P, `pk_`P.

In order to increase the trust on the key validity, the Consumer can request the key to more than one notary, e.g. N_S, and wait for the answer of N_T notaries, where $N_T \leq N_S$. Moreover, we assume that the notaries are neighbor nodes and that could be part of different network areas in order to guarantee partition tolerance. If fewer than N_T of notaries answer to the Interest within a time interval Δt, the Consumer requests the key to the TA using the nonce-based method.

It is worth noting that the notaries sign the keys with their keys, which, therefore must be available at the Consumer or can be retrieved with one of the other protocols.

Algorithm 4 describes the operations performed by the Consumer.

Algorithm 4 Consumer in protocol P4.

```
Send I(notary/$N_i$/key/pk_$P$) to the chosen notaries.
Wait $\Delta t$.
if At least $N_T$ D_$N_i$(notary/$N_i$/key/pk_$P$) packets are received then
    Check that the key signatures are valid.
    Store the key in the CS.
else {$\Delta t$ expires}
    Retrieve /key/pk_$P$ using the nonce-based method (Protocol 2).
end if
```

Notary nodes store a *key entry* for each Producer, P. The entry is a special content whose name is `/key/pk_`P, and it is composed of the key itself and the corresponding lifetime or the label *revoked*. The lifetime is generated by the key owner when the key is first used, and timely updated. When the key's lifetime expires or the key becomes revoked, the key is marked as stale and is automatically dropped out of the cache.

The Notary can answer to the requesting Consumer with two messages: (i) the *key entry*, meaning that the Notary knows the key and it is not expired; (ii) the *key entry* with the label *revoked*, meaning that it is no longer valid to sign and must be dropped. The Data packet has `name=notary/`N_i`/key/pk_`P. The signed info is the Publisher Public Key Digest that identifies the Notary that has signed the content and the content is the *key entry* or the *key entry* with the label *revoked*. The packet is signed with the Notary key.

The following Algorithm 5 describes the operations performed by the Notary.

This solution allows the Consumer to choose where to anchor its trust. Moreover, we do not create a bottleneck in the Trusted Authority, since each Notary can sign and forward the key entry.

Algorithm 5 Notary N_i in protocol P4.

```
Receive I(notary/$N_i$/key/pk_$P$).
Remove name prefix and check Content Store.
if D(/key/pk_$P$) is found then
  if key lifetime != 0 && key is not revoked then
    send D_$N_I$(notary/$N$/key/pk_$P$) to the Consumer.
  else
    if key is revoked then
      send D_$N_i$(notary/$N$/key/pk_$P$) with label revoked
    end if
  end if
end if
```

We only involve the TA when there is a new key request or when the key expires. Also, the key lifetime checking does not require a coordination between the nodes clock.

This key retrieval scheme is secure if the security window is $W > \Delta t + RTT$, i.e. it should be bigger than the waiting time window plus the round trip time between the Consumer and the TA if the Notaries do not answer within Δt. Usually, the security window, W, coincides with the key lifetime that is assigned to the key and signed by the TA, when the key is created.

6. Performance evaluation

In this section we evaluate the number of exchanged messages as a function of the system parameters $|P|$, $|R|$, $|C|$, and $|N_T|$ of the protocol presented in Section 3.3.

First, it is useful to discuss the possible size of the packets. As stated in [23], the minimum size of the Interest packet is 14 byte, let us call it mS_I. The maximum size is 196,619 byte, let us represent it as MS_I. Thus, the minimum size of the Data packet is 6 byte, mS_D, the maximum size is 196,611 byte, MS_D. Then, we consider the frequency of requests λ, measured as the number of sent interests per second. Moreover, we define the miss probability at router node r, $p^r_{miss} = \prod_{i \in Path_{C \to r}} (1 - p^i_{hit})$, where p^r_{hit} is the hit probability at router node r, $r \in Path_{C \to r}$ comprises all the router nodes on the path between the Consumer C and the Router r. Then, we define the stale probability, p^r_{stale}, that is the probability that the key timestamp stored in the Router r is expired.

6.1. Number and size of messages

During the **Key Gen** phase, the pth Producer generates his key pair and pushes the public key to the Trusted Authority, that receives $|P|$ data packets. Then, the TA certifies the public keys and sends them back to the corresponding pth Producer. Finally, the Trusted Authority pushes all the Producers' public keys to all the $|C|$ Consumers.

The **Send Interest** phase involves only the Consumers. The cth Consumer sends λ Interest packets per seconds to the neighbor Routers.

During the **Forward Interest** phase, each rth Router receives $\frac{\lambda|C|}{|R|} p^{r-1}_{miss}$ Interests and forwards to the next hops $\frac{\lambda|C|}{|R|} p^r_{miss}$ Interest packets.

The **Send Data** phase can include a Producer or a Router. The Router node receives $\frac{\lambda|C|}{|R|} p^{r-1}_{miss}$ Interest packets and answers with $\frac{\lambda|C|}{|R|} p^{r-1}_{miss}$ Data packets. The Producer node receives $\lambda|C|p^r_{miss}$ and answers with the same number of Data packets.

The cth Consumer receives λ Data packets from the neighbor nodes during the **Receive Data** phase.

Finally, each Consumer verifies the Data packets received and checks the key validity in the **Verify Data** phase. The number of messages is different relative to the freshness protocol chosen. Starting from Protocol 1, for each key update, the TA pushes into nodes repositories the Producers' public key. Thus, it sends $|C||P|$ Data packets and each Consumer receives $|P|$ Data packets containing the Producer's public key.

Using the nonce-based method, P2, the cth Consumer sends an Interest for the public key for each Producer. The rth Router receives $|C||P|/|R|$ Interests and forwards them to the next hop until they reach the TA. The TA answers with $|C||P|$ Data packets that are forwarded by the Routers till the cth Consumer.

By choosing the timestamp-based Protocol 3, the cth Consumer sends an Interest for each Producer's key specifying the time threshold that the key should not exceed. Each r Router receives $\frac{|C||P|}{|R|} p^{r-1}_{stale}$ Interests and checks their timestamp. If the keys are not stale, the Router answers with the corresponding $\frac{|C||P|}{|R|} p^{r-1}_{stale}$ Data packets. Otherwise, the Router forwards $|C||P|/|R|p^r_{stale}$ Interests to the next hop. The Interests can reach the TA, that sends on the reverse path the corresponding keys. Finally, the $|P|$ keys are received by the cth Consumer.

Concluding with Protocol 4, the cth Consumer sends $|N_T|$ Interests for each Producer's key to the chosen Notaries. Each of the chosen nth Notary responds with the corresponding Data packet. If the Notary does not have the key or the key is stale, the Consumer sends an Interest using the Protocol 2.

We now evaluate the size of messages: Interest and Data packets. We follow the specifications on [23] for our analysis. Our Interest packets can have two names: `/data/Id_P/seq_no` or `/key/Id_P/seq_no`. Thus, the Interest name has different sizes depending on the packet type: $S_I(\texttt{/data/Id\_P/seq\_no}) = mS_I + L(PPKD) + L(name) = 286$ byte or $S_I(\texttt{/key/Id\_P/seq\_no}) = mS_I + L(PPKD) + L(name) = 280$ byte, where $L(PPKD) = 256$ byte, and $L(name)$ depends on the name length. Moreover, when the nonce-based method is used, the *Selector* field should be used for specifying the `AnswerOriginKind` that requires 3 byte more. When the timestamp-based method is used, the name comprises also the timestamp that requires 22 byte. If we use the Protocol 4, we need to add `/notary/Id_N` to the name and it is 11 byte more.

We can state the same about the Data packet names, i.e. the length depends on the protocol chosen. Usually, the content size changes depending on the contents, however, we assume that all the contents (i.e. data and keys) have the same size of 128 byte. Thus, the Data packet size is $S_D(\texttt{/data/Id\_P/seq\_no}) = mS_D + L(sign) + L(content) + L(name) =$ 676 byte or $S_D(\texttt{/key/Id\_P/seq\_no}) = mS_D + L(sign) + L(content) + L(name) = 670$ byte, where $L(sign) = 260$ byte, $L(content) = 396$ byte, and $L(name)$ depends on the name length.

Table 1 compares the number of messages received and sent by each entity and reports the corresponding message sizes.

6.2. Assessment scenario

To evaluate the impact of key retrieval on NDN nodes, we conduct simulations using the open-source *ndnSIM* package [24], which implements NDN protocol stack for NS-3 network simulator. We run simulations for two network topologies: i) a smaller tree topology, and ii) a larger mesh topology. The nodes' Content Stores use the Least Recently Used (LRU) cache replacement policy, the link between each pair of nodes is bidirectional and has a capacity of 1 Gbit/s and a latency of 5 ms.

The tree topology, in Fig. 5, comprises 8 leaf nodes, the Consumers, 6 Router nodes that are the transit nodes organized into two levels and one root node, the Producer.

The mesh topology, in Fig. 6, comprises 13 leaf nodes, the Consumers, 13 Router nodes and 13 root nodes, the Producers.

The Consumer nodes alternates On and Off periods following an exponential distribution. During On period, the Consumer requests

Table 1
Message exchange.

Phase	Input Messages	Output Messages	Size in (byte)	Size out (byte)
		Trusted Authority		
Key Gen	$\|P\|$	$\|P\| + \|C\|\|P\|$	670	670
Verify P1	–	$\|C\|\|P\|$	–	670
Verify P2	$\|C\|\|P\|$	$\|C\|\|P\|$	283	670
Verify P3	$\|C\|\|P\|p_{stale}^{r-1}$	$\|C\|\|P\|p_{stale}^{r-1}$	303	692
		Producer		
Key Gen	1	1	670	670
Send Data	$\lambda\|C\|p_{miss}^{r}$	$\lambda\|C\|p_{miss}^{r}$	286	676
		Router		
Forward Int	$\frac{\lambda\|C\|}{\|R\|} p_{miss}^{r-1}$	$\frac{\lambda\|C\|}{\|R\|} p_{miss}^{r}$	286	286
Send Data	$\frac{\lambda\|C\|}{\|R\|} p_{miss}^{r-1}$	$\frac{\lambda\|C\|}{\|R\|} p_{miss}^{r-1}$	286	676
Verify P2	$\frac{2\|C\|\|P\|}{\|R\|}$	$\frac{2\|C\|\|P\|}{\|R\|}$	953	953
Verify P3	$\frac{2\|C\|\|P\|}{\|R\|} p_{stale}^{r-1}$	$\frac{2\|C\|\|P\|}{\|R\|} p_{stale}^{r-1}$	995	995
		Consumer		
Key Gen	$\|P\|$	–	670	–
Send Int	–	λ	–	286
Receive	λ	–	676	–
Verify P1	$\|P\|$	–	670	–
Verify P2	$\|P\|$	$\|P\|$	670	283
Verify P3	$\|P\|$	$\|P\|$	692	303
Verify P4	$\|N_T\|\|P\|$	$\|N_T\|\|P\|$	684	294
		Notary		
Verify P4	$\|C\|\|P\|$	$\|C\|\|P\|$	294	684

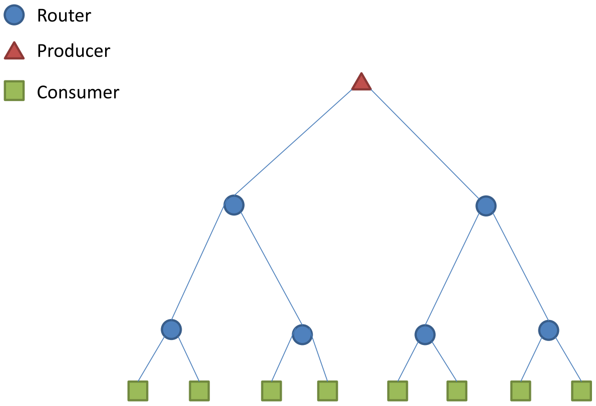

Fig. 5. Tree topology.

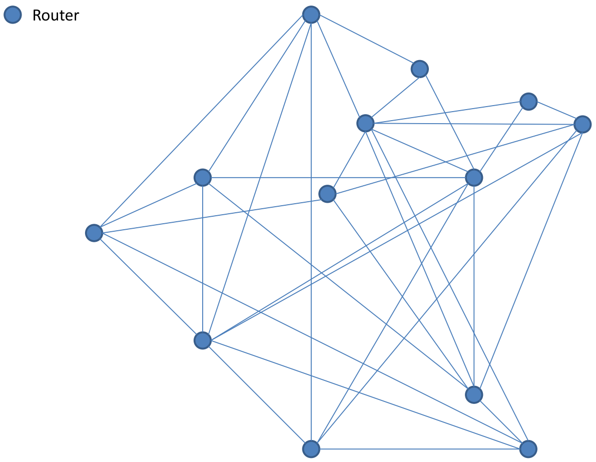

Fig. 6. Mesh topology. Each Router has a link with a Consumer and a Producer, not shown in the picture.

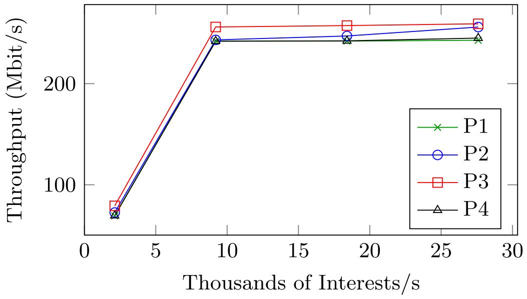

Fig. 7. Mean volume of Data packet received by the Consumer nodes in the tree topology.

content with a frequency f_c interests/s, with an exponential distribution with mean $1/f_c$ for inter-interests gap.

The contents of each Producer are organized into $C = 400$ popularity classes, each one with 34,500 objects. Each content, c, has a probability of being requested, p_c, that follows the Zipf-Mandelbrot law: the more the content is popular, the higher the probability of being requested, hence $p_c = K(c+q)^{-\alpha}$, where $K = 1/\sum_{i=1}^{C}(i+q)^{-\alpha}$ and $\alpha = 0.8$ is the slope of the distribution. Since we put $q = 0$, the distribution becomes the Zipf law.

We assume that the number of Producers is equal to the number of Consumers. Thus, eight Producer applications are co-located in the root node in the tree topology. The Producer are distributed in the network as depicted in Fig. 6 for the mesh topology. We also assume that the TA responsible for signing a Producer key is co-located with the Producer.

All the Routers and Consumers have a Content Store that allows up to 207,000 entries to be cached and respects the content freshness. The simulations last 200 s. All the results are averaged over 10 simulations achieving a confidence interval of 95% or higher that yields a precision better than 1% .

We fixed unlimited freshness for "/data" packets and limited freshness for "/key" packets. Further, to distinguish between the nonce-based, the timestamp-based and the proactive methods, we assume the following:

1. in the proactive mode (P1), the Consumers always have the necessary keys;
2. in the nonce-based protocol (P2), the routers do not store keys;
3. in the timestamp-based protocol (P3), the validity threshold is chosen 10 s before the current time;
4. in the notary-based method (P4), the key validity is 10 s, and routers do not store keys;
5. in P2, P3, and P4, the TA signing Producer P keys is co-located with the Producer itself;
6. in P4, each Consumer chooses the three nearest Consumers as notaries and waits for the first answer.

6.3. Numerical results

In this Section, we compare the performance of the protocols discussed in this paper. In particular, we report data concerning the latency depending on volume of requests, and the throughput on network links by distinguishing between tree and mesh topology.

Figs. 7 and 8 show the average throughput on the input link of the Consumer nodes for the different key retrieval methods for the two topologies. As can be noted in Fig. 7, in the tree topology the throughput on the link between the first level of Routers and the Consumer nodes grows with the frequency of requests and reaches saturation at about 250 Mbit/s, which is much less than the channel capacity. In this topology, the link at the Producer node is the bottleneck. The throughput depicted in the Figure takes into account both the data content and the key content, where data con-

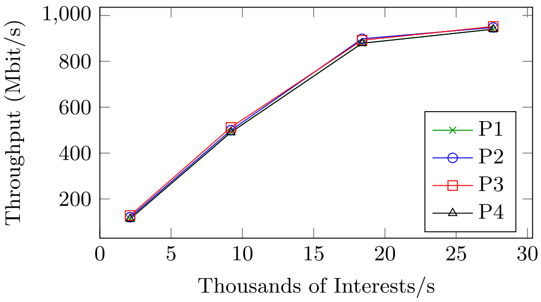

Fig. 8. Mean volume of Data packet received by the Consumer nodes in the mesh topology.

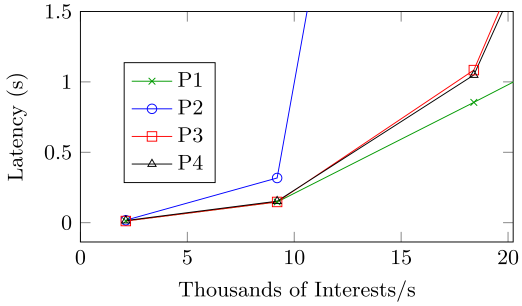

Fig. 9. Mean latency of all the popularity classes depending on volume of requests in the tree topology considering the alternative protocols. Precision better than 1% with confidence 95%.

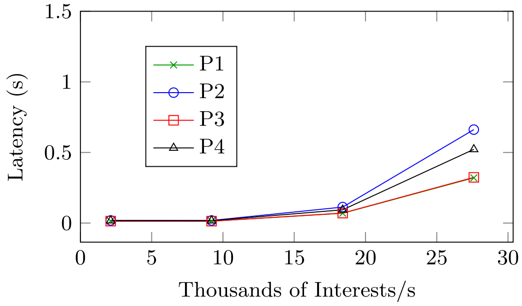

Fig. 10. Mean latency of all the popularity classes depending on the volume of requests in the mesh topology considering the alternative protocols. Confidence 95%.

tent occupies most of the available channel capacity. The impact of key packets is negligible with respect to the data content and, for this reason, all the protocols show similar throughput trend. Even if more interests for keys are sent, e.g., in P2 protocol, they do not have a perceivable effect on the throughput.

A similar trend is depicted in Fig. 8, which is relative to the mesh topology. Differently from the tree topology, there is no bottleneck at the Producer node and the throughput approaches the channel capacity, i.e. 1 Gbit/s. As in the tree topology, the impact of the key retrieval protocol is negligible on the throughput and all protocols show a similar throughput, this is due to fact that the data content occupies most of the available capacity with respect to the key packets.

Figs. 9 and 10 show the mean latency for content and key retrieval averaged over all the popularity classes as a function of the frequency of requests, i.e. f_c = 2123, 9200, 18,400, 27,600 interests/s, and depending on the network topology. In the tree topology, Fig. 9, the lowest latency is obtained with P1, which grows slowly and stays under 0.3 s even with f_c = 27,600. Protocol P3 and P4 show similar latency as P1 up to f_c = 18,400, when their delay starts growing, reaching about 1 s for f_c=27,600. Finally, protocol P2 latency is comparable to the other methods only up to $f_c = 9200$, then the latency starts growing quickly, showing that P2 performance suffers from the congestion at the root node.

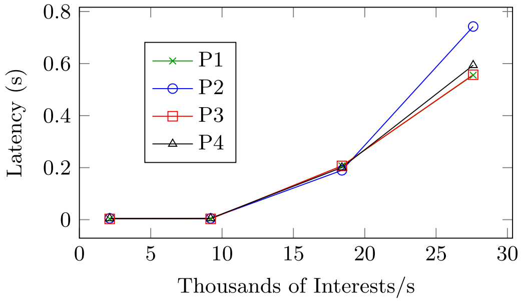

Fig. 11. Standard deviation relative to the mean latency of all the popularity classes depending on the volume of requests in the mesh topology considering the alternative protocols.

We have not taken into account a congestion control mechanism. Thus, in Fig. 9, the values of latency relative to the biggest volumes of requests grow exponentially due to the congestion of the links. Before the channel saturation, in the left-hand side of the Figure, the additional latency of the reactive protocols P2, P3, and P4, is negligible with respect to the proactive protocol, P1. The proactive protocol can be seen as the lower bound for the considered topologies. Indeed, since the keys are updated out-band, the depicted values represent the latency only in content retrieval.

The results relative to the mesh topology are presented in Fig. 10. The trend is similar to the tree topology, but with some interesting differences. First, the average latency is lower for the same f_c, because the traffic can flow over multiple routes and the producer link stops being a bottleneck. Second, the performance of protocol P4 is worse than P3 and halfway between P2 and P3, whereas in the tree topology the performance of P3 and P4 are similar. In fact, in the tree topology, P4 messages flow through the peripheral links, which are not congested. Instead, in the mesh topology, congestion may occur at any link and, thus, involve P4 messages. Additionally, P4 sends two or more interests for each requested key, thus increasing the traffic when compared to P3.

The same observations drawn for the tree topology can be written here. The results relative to P2, P3 and P4 add a negligible latency to the lower bound, P1, before the channel saturation. In this case, the saturation is asymptotically reached with a bigger volume of requests than the tree topology. Thus, we can say that, in both scenarios, our proposed solutions guarantee the benefits of NDN in terms of small latencies in content delivery.

Fig. 11 depicts the standard deviation relative to the mean latency for content and key retrieval averaged over all the popularity classes as a function of the frequency of requests, i.e. f_c = 2123, 9200, 18,400, 27,600 interests/s, and relative to the mesh topology. The figure shows that the standard deviation results are clustered closely around the values of the mean latency. Thus, the curves in Fig. 11 have the same trend of those in Fig. 10.

All our proposed solutions prevent nodes accepting a corrupted packet in their cache, except that less than a time W has passed since the key was revoked. However, observing the results, there are some differences between the presented protocols.

P2 always guarantees keys' authenticity at the price of higher latency and of possible bottlenecks on the Trusted Authority node(s).

P1 has the lowest latency in all cases. The drawback is that P1 requires key pre-distribution, resulting in large memory consumption in the nodes and in longer vulnerability periods. Further, we believe that the proactive mode can be used when the channel capacity is limited and the network is overloaded, since it allows to manage the key retrieval off-line.

P3 has comparable latency to P1 in the mesh topology and similar performance in the tree topology. In addition, P3 can provide much shorter security windows and allows the user to choose the

Table 2
Comparison of the studied protocols.

Protocol P1	
Minimum achievable security window (W)	long
Additional traffic	2 packets per hop to the TA for each key in the database for each Consumer
Key retrieval latency	negligible
Protocol P2	
Minimum achievable security window (W)	1 RTT to the TA
Additional traffic	2 packets per hop to the TA for each key request
Key retrieval latency	1 RTT to the TA
Protocol P3	
Minimum achievable security window (W)	1 RTT to the TA
Additional traffic	between 2 packets and 2 packets per hop to the TA, depending on caching, for each key request
Key retrieval latency	between 1 RTT to first hop and 1 RTT to the TA, depending on caching
Protocol P4	
Minimum achievable security window (W)	1 RTT to the TA + processing time at the notaries
Additional traffic	2 packets per hop to each notary for each key request, plus 2 packets per hop to the TA, in case of failure
Key retrieval latency	1 RTT to the farthest notary plus 1 RTT to the TA, in case of failure

time threshold of validity. It is therefore a viable choice for wide scale deployment. However, it pays the price of clock synchronization of all the network nodes.

Finally, P4 is the best solution not only for a good trade off between the evaluated performance parameters but also because it leaves the users the possibility to choose where to put their trust. Table 2 shows a comparison of the main features of the different protocols in terms of the minimum security window that can be guaranteed, the resulting additional traffic, and the latency on key retrieval.

7. Conclusion

This paper deals with the problem of content freshness and revocation in ICN scenario. In particular, it compares different centralized, P1 and P2, and distributed, P3 and P4, approaches to distribute up-to-date keys in a NDN-friendly way.

We provide the NDN framework with a trust management infrastructure. However, our approach can be straightforwardly extended to other ICN approaches. In particular, we present a proactive method that periodically distributes updated keys to the nodes, two reactive protocols that allow the TA or an intermediate node to send the up-to-date status of the key upon request, and a method where some trusted nodes provide keys on behalf of the TA. Our results show that, even if the communication model undergoes a change, it is possible to maintain the benefits of an NDN network in terms of latency.

On the one hand,both P1 and P2 are centralized methods because only one node, the Trusted Authority, can provide the requested key. These methods can be used when the access network is overloaded, however, the TA could become a bottleneck. To overcome this issue, a key delegation scheme could be defined to allow a TA to entrust some other network entities to certify keys. This is a possible direction for an evolution of our work.

On the other hand, both P3 and P4 are distributed methods because any cache in the network can send the relevant key. The difference is that P4 further reduces the load towards the Trusted Authority and makes the system more robust to network partitioning. The advantages of P4 come at the expense of additional peer-to-peer traffic in the network. As such, P4 works best when the bottleneck is farther from the user, such as in our mesh topology. When the bottleneck link is near to the user, a solution with no peer-to-peer traffic would have better performance.

References

[1] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, T. Waehlisch, Icn research challenges, 2015,.https://tools.ietf.org/html/draft-irtf-icnrg-challenges-01.

[2] E. AbdAllah, H. Hassanein, M. Zulkernine, A survey of security attacks in information-centric networking, Commun. Surv. Tutorials, IEEE 17 (3) (2015) 1441–1454, doi:10.1109/COMST.2015.2392629.

[3] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile, 2008, RFC 5280.

[4] M. Myers, R. Ankney, M. A., S. Galperin, C. Adams, X.509 internet public key infrastructure,online certificate status protocol - ocsp, 1999. RFC 2560.

[5] D. Wendlandt, D.G. Andersen, A. Perrig, Perspectives: Improving ssh-style host authentication with multi-path probing, in: USENIX 2008 Annual Technical Conference on Annual Technical Conference, in: ATC'08, USENIX Association, Berkeley, CA, USA, 2008, pp. 321–334.

[6] D. Smetters, V. Jacobson, Securing Network Content, Technical Report, PARC, 2009.

[7] Y. Yu, Public Key Management in Named Data Networking, Technical Report, UCLA, April, 2015.

[8] C. Ghali, G. Tsudik, E. Uzun, Network-layer trust in named-data networking, SIGCOMM Comput. Commun. Rev. (2014) 12–19.

[9] P. Mahadevan, E. Uzun, S. Sevilla, J. Garcia-Luna-Aceves, Ccn-krs: a key resolution service for ccn, in: Proceedings of the 1st International Conference on Information-centric Networking, in: INC '14, ACM, New York, NY, USA, 2014, pp. 97–106.

[10] J. Muoz, J. Forn, O. Esparza, M. Soriano, Cervantes a certificate validation test-bed, in: S. Katsikas, S. Gritzalis, J. Lpez (Eds.), Public Key Infrastructure, Lecture Notes in Computer Science, 3093, Springer Berlin Heidelberg, 2004, pp. 28–42.

[11] E. Syta, I. Tamas, D. Visher, D.I. Wolinsky, B. Ford, Decentralizing authorities into scalable strongest-link cothorities, CoRR abs/1503.08768 (2015). http://arxiv.org/abs/1503.08768.

[12] P. Gasti, G. Tsudik, E. Uzun, L. Zhang, Dos and ddos in named data networking, in: Computer Communications and Networks (ICCCN), 2013 22nd International Conference on, 2013, pp. 1–7.

[13] A. Compagno, M. Conti, P. Gasti, G. Tsudik, Poseidon: Mitigating interest flooding ddos attacks in named data networking, in: Local Computer Networks (LCN), 2013 IEEE 38th Conference on, 2013, pp. 630–638.

[14] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, L. Zhang, Interest flooding attack and countermeasures in named data networking, 2013,

[15] C. Ghali, G. Tsudik, E. Uzun, Needle in a haystack: mitigating content poisoning in named-data networking, in: Proceedings of NDSS Workshop on Security of Emerging Networking Technologies (SENT), 2014.

[16] G. Piro, L.A. Grieco, G. Boggia, P. Chatzimisios, Information-centric networking and multimedia services: present and future challenges, Trans. Emerg. Telecommun. Technol. 25 (4) (2014) 392–406.

[17] P.A.R. Center, Ccnx signature generation and verification, 2009, http://www.ccnx.org/releases/ccnx-0.1.2/doc/technical/SignatureGeneration.html.

[18] C. Bian, Z. Zhu, A. Afanasyev, E. Uzun, L. Zhang, Deploying Key Management on NDN Testbed, Technical Report, UCLA, Peking University and PARC, Feb, 2013.

[19] Y. Yu, A. Afanasyev, D. Clark, k. claffy, V. Jacobson, L. Zhang, Schematizing and Automating Trust in Named Data Networking, Technical Report, UCLA, MIT and CAIDA, June, 2015.

[20] M. Mosko, CCNx 1.0 Collection Synchronization, Technical Report, Palo Alto Research Center, Inc., 2014. http://www.ccnx.org/pubs/hhg/4.7\%20CCNx%201.0%20Collection%20Synchronization.pdf.

[21] G. Mauri, G. Verticale, Distributing key revocation status in named data networking, in: Advances in Communication Networking, Springer, 2013, pp. 310–313.

[22] Z. Zhu, A. Afanasyev, Let's chronosync: decentralized dataset state synchronization in named data networking, in: 21st IEEE International Conference on Network Protocols (ICNP 2013), 2013.

[23] A. Afanasyev, I. Moiseenko, L. Zhang, ndnsim packet format, 2011–2013, http://ndnsim.net/ndnsim-packet-formats.html.

[24] A. Afanasyev, I. Moiseenko, L. Zhang, ndnSIM: NDN Simulator for NS-3, Technical Report NDN-0005, NDN, 2012. http://named-data.net/techreports.html.

Giulia Mauri is a PhD student in Telecommuncation Engineering at the Department of Electronics, Information and Bioengineering of Politecnico di Milano. Her research is focused on Information Centric Networking, in particular her works cover problems related to trust management and privacy. Moreover, Giulia has defined optimization models for mobile scenarios. Giulia is enrolled in the Scuola Interpolitecnica project, that offers a joint PhD program of high qualification. Giulia received the Master of Science Degree “cum laude” in Telecommunication Engineering in October 2012, the thesis was about preserving privacy in a Smart Grid scenario, and Giulia received the 2nd price from CLUSIT for her thesis.

Giacomo Verticale is assistant professor at Politecnico di Milano, Italy. He obtained his PhD in Telecommunication Engineering in year 2003 from Politecnico di Milano defending a thesis on the performance of packet transmission in UMTS. In the years 1999–2001 he was with the research center CEFRIEL, working on the Voice-over-IP and ADSL technologies. He was involved in several European research projects advancing the Internet technology. His current interests focus on the security issues of the Smart Grid and on Network Function Virtualization. He is a member of IEEE and ACM.