ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
School of Information Sciences and Technology
Department of Computer Science

Information-Centric Networking: Security Requirements and Solutions

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Nikos Fotiou

Athens
June 2014

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my advisor Prof. George C. Polyzos for his patient guidance and enthusiastic support. It has been a great honor working with him.

I am particularly grateful for the assistance given by Assistant Professors George Xylomenos, Giannis F. Marias, and Vasilios A. Siris. Their useful and valuable comments helped me progress.

I am very thankful to Manos Dramitinos, for encouraging and helping me to pursue my PhD studies.

I had the privilege of working at the Mobile Multimedia Lab, with great colleagues and friends. Konstantinos Katsaros, Christoforos Ververidis, and Pantelis Fragoudis were the first persons I met there. They were very supportive. Working with them was a pleasure and even when they graduate they were always available for helping me. Vaggelis G. Douros (the 'director'), Haris Stais, Christos Tsilopoulos, and Xenofon Vasilakos joined the MMLab almost at the same time as me. Together we had endless conversations and great fun, inside and outside the lab. I will always be thankful for meeting them. Yiannis Thomas belongs to the new generation. He is also a great friend and I wish him the best.

I would like also to extend my thanks to the members of the Network Economics and Services Laboratory, as well as, to Nausika Kokkini, Kostas Makedos, and Vasilis Kemerlis.

I am grateful to Nikos Ioannou and Yiannis Tsokos for their support during my first months in Athens. I also thank Dimitris Tsoutsias, Argiris Orfanos, Antonis Glykas, Christos Angelopoulos, Charalampos Doukas, Ilias Begetis, and Stefanos Nikolidakis.

Last, but surely not least, I would like to deeply thank my family and my wife Rea for their invaluable support.

VITA

2005	Dipl. in Information and Communication Systems Engineering, University of the Aegean, Greece
2007	M. Sc. in Internetworking, KTH Royal Institute of Technology, Sweden
2014	Ph. D. in Computer Science, Athens University of Economics and Business, Greece

PUBLICATIONS

Journal Publications

P. Zhang, N. Fotiou, B. E. Helvik, G. F. Marias, and G. C. Polyzos, "Analysis of the effect of InfoRanking on content pollution in P2P systems," in *Wiley Security and Communication Networks*, vol. 7, no. 4, pp. 700-713, 2014

C. Doukas, N. Fotiou, G. C. Polyzos, and I. Maglogiannis, "Cognitive and Context Aware Assistive Environments Using Future Internet Technologies," in *Springer Universal Access in the Information Society*, vol. 13, no. 1, pp. 59-72, 2014

G. Xylomenos, C. Ververidis, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, G. C. Polyzos, "A Survey of Information-Centric Networking Research," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024-1049, 2013

N. Fotiou, K. Katsaros, G. C. Polyzos, M. Sarela, D. Trossen, G. Xylomenos, "Handling Mobility in Future Publish-Subscribe Information-Centric Networks," in *Springer Telecommunication Systems, Springer, Special Issue on Mobility Management in the Future Internet*, vol. 53, no. 3, pp. 299-314, 2013

N. Fotiou, G. F. Marias, and G. C. Polyzos, "Access Control Enforcement Delegation for Information-Centric Networking Architectures," in *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 497-502, 2012 (selected for CCR publication as best paper of ACM SIGCOMM Workshop on Information-Centric Networking 2012)

N. Fotiou, G. C. Polyzos, and D. Trossen, "Illustrating a Publish-Subscribe Internet Architecture," in *Springer Telecommunication Systems*, vol. 51, no. 4, pp. 233-245, 2012

Book Chapters

N. Fotiou, G. F. Marias, and G. C. Polyzos, “Publish-Subscribe Internetworking Security Aspects,” in *Trustworthy Internet*, G. Bianchi, L. Salgarelli, N. Blefari-Melazzi, eds., Springer, pp. 3-15, 2011

Refereed International Conference and Workshop Papers

M. Giannikos, K. Kokoli, N. Fotiou, G. F.Marias and G. C. Polyzos, “Towards Secure and Context-Aware Information Lookup for the Internet of Things,” in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC 2013)*, San Diego, CA, USA, 2013

G. F. Marias, N. Fotiou, G. C. Polyzos, “Efficient information lookup for the Internet of Things, ” in *IEEE WoWMoM: Proceedings of the 1st Workshop on the Internet of Things: Smart Objects and Services*, San Francisco, CA, USA, June 2012

N. Fotiou, G. F. Marias, G. C. Polyzos, “Fighting phishing the information-centric way,” in *Proceedings of the 5th IFIP International Conference on New Technologies, Mobility and Security (NTMS 2012)*, Istanbul, Turkey, May 2012

K. V. Katsaros, N. Fotiou, X. Vasilakos, C. N. Ververidis, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, “On Inter-domain Name Resolution for Information-Centric Networks,” in *Proceedings of the IFIP Networking 2012*, Prague, Czech Republic, May 2012

N. Fotiou, G. F. Marias, G. C. Polyzos, P. Szalachowsk, Z. Kotulski, M. Niedermeier, X. He, and H. de Meer, “Towards Adaptable Security for Energy Efficiency in Wireless Sensor Networks,” in *Proceedings of the 28th Wireles World Research Forum (WWRF) Meeting*, Athens, Greece, April 2012

C. Doukas, N. Fotiou, G. C. Polyzos, and I. Maglogiannis, “Context-Aware Delivery of Information in Assistive Environments utilizing Future Internet Technologies,” in *Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, Crete, Greece, May 2011

N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, “Developing Information Networking Further: From PSIRP to PURSUIT,” in *Proceedings of the 7th International ICST Conference on Broadband Communications, Networks, and Systems*, Athens, Greece, October 2011

N. Fotiou, G. F. Marias, and G. C. Polyzos, “Security Aspects of a Clean Slate Information Oriented Internet Architecture,” in *Proceedings of the 21st International Tyrrhenian Workshop on Digital Communications*, Ponza, Italy, September 2010

——, "Towards a Secure Rendezvous Network for Future Publish/Subscribe Architectures," in *Proceedings of the 3rd Future Internet Symposium*, Berlin, Germany, September 2010.

——, "Fighting Spam in Publish/Subscribe Networks Using Information Ranking" in *Proceedings of the 6th Euro-NF Conference on Next Generation Internet (NGI)*, Paris, France, June 2010.

——, "Information Ranking in Content-Centric Networks," in *Proceedings of Future Network and Mobile Summit 2010*, Florence, Italy, June 2010.

N. Fotiou, G. C. Polyzos, and D. Trossen, "Illustrating a Publish-Subscribe Internet Architecture," in *Proceedings of the Euro-NF: Future Internet Architecture - New Trends in Service and Networking Architectures, 2nd workshop*, Santander, Spain, June 2009.

K. Katsaros, N. Fotiou, G. C. Polyzos, and G. Xylomenos, "Overlay multicast assisted mobility for future publish/subscribe networks," in *Proceedings of the 18th ICT Mobile Summit*, Santander, Spain, June 2009.

——, "Supporting mobile streaming services in future publish/subscribe networks," in *Proceedings of the Wireless Telecommunications Symposium (WTS 2009)*, April 2009.

ABSTRACT OF THE DISSERTATION

Information-Centric Networking:
Security Requirements and Solutions

by

Nikos Fotiou

Doctor of Philosophy in Computer Science

Athens University of Economics and Business, Athens, 2014

Professor George C. Polyzos, Chair

Information-Centric Networking (ICN) is an enticing paradigm receiving increasing attention by the research community due to its intriguing characteristics. ICN based architectures enable the interconnection of content items, rather than network endpoints, i.e., all core (inter-)networking functions are built around information (content) identifiers rather than location identifiers. This networking swift promises to overcome limitations of the current Internet architecture, including inefficient use of resources, lack of effective mobility and multicast support, and weak trust mechanisms that lead to security breaches, such as, Denial of Service (DoS), scamming attacks, and loss of privacy. Although in principle ICN seems to overcome the security problems of the current Internet architecture, when it comes to its realization security concerns are raised: users are exposed to unwanted traffic, content is scattered in multiple network locations, lookups reveal users' actual interests. It has, therefore, become obvious that the security requirements and properties of ICN should be reconsidered and new ICN specific security solutions have to be developed. In this context, we review key ICN research efforts and we propose new security requirements. Moreover we design, implement, and evaluate ICN specific security solutions that provide access control for content stored in multiple locations, preserve users' privacy, and protect users against spam. Finally we show how these security solutions can also be beneficial to the current Internet architecture.

Chapter 1

Introduction

The current Internet architecture cannot effectively and efficiently handle various challenges, including security, mobility, scalability, quality of service, and economics, due to the focus and shortcomings in its original design [1]. The Internet still significantly resembles the telephone network in many design choices–and since its inception this design has been hardly changed. A key goal of the Internet was to efficiently interconnect mainframes and minicomputers, and to provide efficient remote access to them. This end-to-end approach and especially its specific practical implementation, however, have been identified as a root cause of many limitations of the current Internet architecture. Various add-on "patches," such as NATs, Mobile IP, CDNs, P2P overlays, etc., all violate, in various ways, several aspects of the original Internet architecture in order to provide answers to features that were not part of the original design (or the original requirements). Moreover, the original Internet architecture and protocols were developed assuming a legitimate and cooperative environment, which is far from today's reality, where competition is widespread and lack of trust and security threats, such as malware, Denial of Service (DoS) attacks, and phishing, have become more and more prevalent.

More recently, it has been observed that information access has become the typical Internet use, e.g., as manifested by the majority of the reaffic being associated with it [2]. This information-centric usage of the Internet raises various architectural challenges, many of which are not effectively handled by the current architecture [3]. These challenges include medium-independent information access, tussle mediation through information governance, privacy and accountability through controlled information dissemination, and information scarcity. To this end, the question of whether we can continue "patching over patches," or whether a new clean-slate architectural approach for the Internet is actually needed [4] has been raised. Along these lines, a research community has been formed which, having identified the limitations of the current Internet, is discussing the key requirements and objectives of the Future Internet, and is proposing new architectures and paradigms to address them.

In this context, *Information-Centric Networking* (ICN) has emerged as a promising candidate for the architecture of the Future Internet. ICN aims to reflect current and future needs better than the existing Internet architecture. By naming content at the network layer, ICN favors the deployment of in-network caching (or storage, more generally) and multicast mechanisms, thus facilitating the efficient and timely delivery of information to the users.

1.1 Motivation for the dissertation

Security has always been one of the flagships of ICN: most ICN research efforts have set as one of their main goals to mitigate the shortcomings of the current Internet architecture. However, it seems that most of these proposals are limited at the inherent security properties of the ICN paradigm.

Indeed, ICN by design has many characteristics that make it less vulnerable to the attacks that overwhelm the current Internet architecture. In ICN, the network is not oblivious to the content being transfered, therefore malicious content can be filtered out, preventing this way the spread of malware and the launch of phishing attacks. Moreover, one of the basic principles of ICN is that no content flow occurs unless there exist explicit signaling denoting the demand, as well as, the availability of a specific content item. This property is expected to decrease unwanted traffic (spam), as well as, DoS attacks. Finally, in an ICN architecture content is expected to be dispersed in many network locations, contributing to the greater availability of the architecture. However, these properties of ICN are a double-edged sword, as they facilitate other forms of attacks. For example, using content identifiers as an enabler of the (inter-)networking functions may jeopardize end-users privacy: by having users requesting content identifiers, rather than network locations, one can easily profile user preferences. Moreover, if an attacker succeeds in making the network believe that some malicious content is legitimate, the network will facilitate the delivery of this piece of content to many users. Finally the dispersion of content in many network locations can lead the content owner in losing control of her content: a content owner can hardly impose on entities, laying outside her administrative realm, how to disseminate the stored content.

These observations, give us a strong motive to: (i) revisit key ICN proposals, (ii) define concrete security requirements and (iii) design new security solutions.

1.2 Contributions

With this work we make the following contributions

- We review DONA [5], NDN [6], PURSUIT [7], SAIL [8], COMET [9], CONVERGENCE [10] and MobilityFirst [11] ICN architectures. We discuss how they implement network functionalities and we present their main design choices with respect to security.

- We specify ICN security requirements related to end-users, content and infrastructure. For each security requirement we discuss related security solutions.

- We design an access control delegation mechanism that enables access control enforcement on content stored outside the administrative realm of its owner. We specify an ICN-based communication protocol and we implement it for the PSI ICN architecture.

- We design a "content-centric" spam protection mechanism for ICN. We argue for its feasibility and we show, through simulations, its efficiency compared to the “end-user”-centric solution proposed by Tarkoma [12].

- We design a privacy preserving information lookup solution. We leverage an existing *Private Information Retrieval* (PIR) solution, proposed by Bethencourt et al. [13], and we achieve significantly better performance in terms of communication overhead, by exploiting the hierarchical organization of the information space.

- We apply our access control delegation solution in the Cloud environment. We show how it can help enterprises to overcome some of the entry barriers of the existing Cloud paradigm. We prove its security and we present an implementation using the *OpenStack* Cloud stack.

- We apply our anti-spam solution in a P2P file sharing system in order to prevent the spread of “polluted” files. We discuss an implementation strategy and we show through simulations, that the fact that our solution supports positive only votes makes it more efficient compared to the Credence object-ranking system [14].

1.3 Dissertation outline

The remainder of the dissertation is organized as follows. Chapter 2 provides a review of some key ICN research efforts and defines a conceptual ICN architecture, the Publish-Subscribe Internetworking (PSI) architecture, which is used as a reference model in the following chapters. Chapter 3 discusses ICN security requirements and presents existing security solutions specifically designed to cope with these requirements. Chapter 4 details our security solutions for *access control*, *spamming* and *privacy*. Chapter 5 discusses the applications of our solutions in the current Internet in the context of Cloud computing and P2P file sharing. Chapter 6 discusses the relation of our work with related approaches and further identifies future research targets. Finally, the conclusions of this dissertation are presented in Chapter 7.

Chapter 2

ICN architectures

In this Chapter we review some key ICN architectures. Part of this chapter is based on the survey of Xylomenos et al. [15].

Generally in ICN a *subscriber* demands some content and a *publisher* supplies it; the network is responsible for mediating supply and demand. ICN architectures consider various desing choices regarding how content is named, located and disseminated.

2.1 DONA

While many projects, starting with TRIAD [16], proposed extending the Internet with content routing capabilities, the *Data Oriented Network Architecture* (DONA) [5] from UC Berkeley is one of the first complete ICN architectures, as it radically changes naming by replacing the hierarchical URLs with flat names. Unlike URLs which are bound to specific locations via their DNS component, the flat names in DONA can be persistent, even if the content moves. This allows content to be cached and replicated at the network layer, thus increasing content availability. Finally, names in DONA allow users to verify that the received content matches a requested name via cryptographic techniques.

In DONA each piece of content (or service) is associated with a *principal.* Names consist of the cryptographic hash of the principal's public key `P` and a label `L` uniquely identifying the content with respect to the principal. Naming

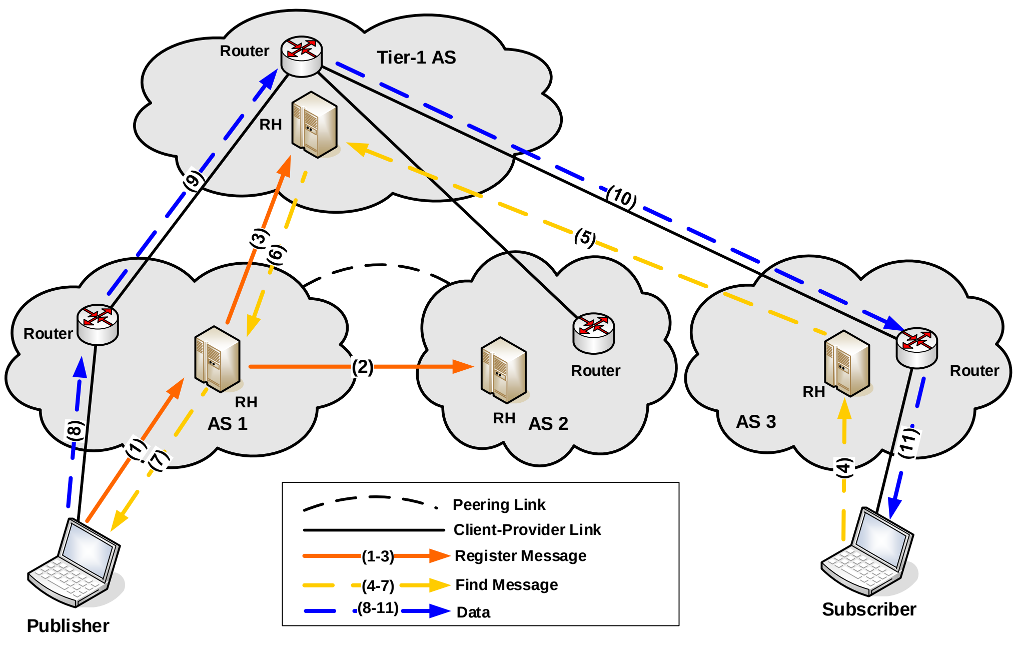


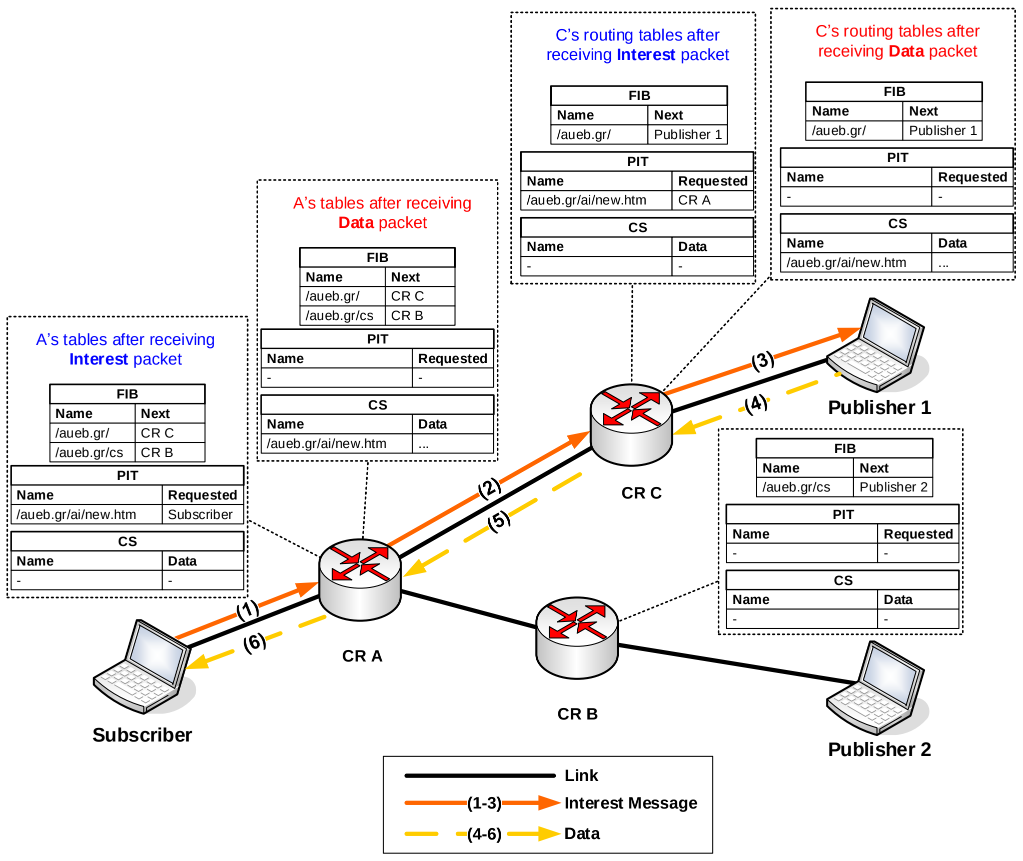
Figure 2.1: The DONA architecture. RH stands for Resolution Handler.

granularity is left to the principals, who are considered to be the owners of the corresponding content. For instance, principals may name either an entire Web site or each individual Web page within it. Names are flat, application-independent, location-independent and globally unique. For immutable data the label can be the cryptographic hash of the content item itself, thus allowing any purveyor (e.g., a CDN) to offer such data. Clients interested in a content are assumed to learn its name through some trusted external mechanisms (e.g., a search engine). Unlike structured DNS names, flat names in DONA do not embed a fixed administrative structure, thus they are easy to map to any private namespace of human-readable names.

A network of specialized servers called *Resolution Handlers* (RHs) is responsible for locating content. There is at least one logical RH at each AS and all RHs are hierarchically interconnected as shown in Figure 2.1. In order to make a content item available, the publisher(principal) sends a REGISTER message with the object's name to its local RH, who stores a pointer to the publisher (arrow 1). The

RH then propagates this registration to the RHs in its parent and peering domains, following the established routing policies (arrows 2-3), causing each intermediate RH to store a mapping between the object's name and the address of the RH that forwarded the registration. As a result, registrations are replicated in RHs all the way up to the tier-1 providers and, since all tier-1 providers are peers with each other, RHs located at tier-1 providers are aware of *all* registrations in the entire network. Publishers can also issue wildcard REGISTER messages to notify the RH hierarchy that they can provide all possible data items for a specific principal. In order to locate an item, a subscriber sends a FIND message to its local RH, which also propagates this message to its parent according to its routing policies, until a matching registration entry is found (arrows 4-5). At that point, requests follow the pointers created by the registrations in order to eventually reach the publihser (arrows 6-7). Since tier-1 providers are aware of all objects in the network, this process is guaranteed to succeed if the requested name exists. Subscribers can also issue wildcard FIND messages to ask for immutable data with a specific label, regardless of its purveyor. FIND messages gather path-labels as they move from RH to RH, indicating the sequence of AS's crossed by the request. When the request reaches the publisher, these path-labels are simply used in the reverse order to retrace the path towards the subscriber (arrows 8-11). DONA can also support multicast channels, by allowing FIND messages to be cached in RHs for a specified period of time and sending content updates in response to these messages until they expire. When additional FIND messages for the same content are received by the RH, they are merged into a single entry with multiple path-labels for the responses, thus creating a multicast distribution tree.

DONA tries to achieve security through content names. Names in DONA are self-certifying, i.e., they allow the subscriber to verify that the data received matches the name requested. For mutable data, a client requesting a content item named `P:L` will also receive as meta-data the public key of the principal (which is bound to `P` via its hash) and a signature for the data object itself, thus allowing the data to be authenticated as coming from the specific principal. For immutable data, the subscriber can simply verify that the label `L` is indeed the cryptographic hash of the content item, regardless of the purveyor acting as the principal. This allows subscribers to choose a purveyor according to its reputation and performance.



A's tables after receiving **Interest** packet

FIB

Name	Next
/aueb.gr/	CR C
/aueb.gr/cs	CR B

PIT

Name	Requested
/aueb.gr/ai/new.htm	Subscriber

CS

Name	Data
-	-

A's tables after receiving **Data** packet

FIB

Name	Next
/aueb.gr/	CR C
/aueb.gr/cs	CR B

PIT

Name	Requested
-	-

CS

Name	Data
/aueb.gr/ai/new.htm	...

C's routing tables after receiving **Interest** packet

FIB

Name	Next
/aueb.gr/	Publisher 1

PIT

Name	Requested
/aueb.gr/ai/new.htm	CR A

CS

Name	Data
-	-

C's routing tables after receiving **Data** packet

FIB

Name	Next
/aueb.gr/	Publisher 1

PIT

Name	Requested
-	-

CS

Name	Data
/aueb.gr/ai/new.htm	...

FIB

Name	Next
/aueb.gr/cs	Publisher 2

PIT

Name	Requested
-	-

CS

Name	Data
-	-

Figure 2.2: The NDN architecture. CR stands for Content Router, FIB for Forwarding Information Base, PIT for Pending Interest Table, CS for Content Store.

2.2 NDN

The *Content Centric Networking* (CCN) [17] architecture from PARC is the other pioneering fully-fledged ICN architecture. Its basic ideas were described in a Google tech talk [2], long before the first paper describing the CCN architecture was published [18]. The *Named Data Networking* (NDN) [6] project, funded by the US Future Internet Architecture program, is further developing the CCN architecture. NDN envisions reshaping the Internet protocol stack by making the exchange of *named data* the thin waist of the Internet architecture, using various networking technologies below the waist for connectivity, including, but not limited to, IP. In NDN a *strategy* layer mediates between the named data layer and the underlying network technologies to optimize resource usage, e.g., to select a link in a multi-homed node, while a *security* layer applies security functionalities directly on named data. A crucial aspect of NDN is that names are hierarchical, thus allowing name resolution and data routing information to be aggregated across similar names, something considered to be critical for the scalability of the architecture.

Names in NDN are hierarchical and *may* be similar to URLs, for example, an NDN name can be `/aueb.gr/ai/main.html`. However, NDN names are *not* necessarily URLs: their first part is not a DNS name or an IP address and they do not have to be human-readable. Instead, in NDN each name component can be anything, including a dotted human-readable string or a hash value. In NDN a request for a name is considered to match any piece of content whose name has the requested name as a prefix, for example, `/aueb.gr/ai/main.html` can be matched by a content item named `/aueb.gr/ai/main.html/_v1/_s1`, which could mean the first segment of the first version of the requested data. After receiving this content item, the subscriber could ask for the next data segment either directly by requesting `/aueb.gr/ai/main.html/_v1/_s2`, or for the next sibling under this version. Alternatively, the subscriber could ask for the next version by requesting the first sibling of `/aueb.gr/ai/main.html/_v1`. While the way content items are segmented is expected to be known by the subscriber's application, the prefix matching rule enables an application to discover what is available. Furthermore, it allows the subscriber to ask for data that have not been produced yet: a publisher can advertise that it can satisfy requests for a specific prefix, and then return content items with complete NDN names. This can be used to implement various applications where content items are generated dynamically, hence their full names cannot be known in advance, such as voice conferencing [19].NDN supports the association of human-readable hierarchical content names with the corresponding content items in a verifiable way [20].

In NDN subscribers issue INTEREST messages to request content items which arrive in the form of DATA messages, with both types of message carrying the name of the requested/transferred content item. As shown in Figure 2.2, all messages are forwarded hop-by-hop by *Content Routers* (CRs), with each CR maintaining three data structures: the *Forwarding Information Base* (FIB), the *Pending Interest Table* (PIT) and the *Content Store* (CS). The FIB maps content names to the output interface(s) that should be used to forward INTEREST messages towards appropriate data sources. The PIT tracks the incoming interface(s) from which pending INTEREST messages have arrived, i.e., those INTEREST messages for which matching DATA messages are expected. Finally, the CS serves as a local cache for content items that have passed through the CR. When an INTEREST arrives, the CR extracts the content name and looks for a content item in its CS whose name matches the requested prefix. If something is found, it is immediately sent back through the incoming interface in a DATA message and the INTEREST is discarded. Otherwise the router performs a longest prefix match on its FIB in order to decide towards which direction this INTEREST should be forwarded. If an entry is found in the FIB, the router records the INTEREST's incoming interface in the PIT and pushes the INTEREST to the CR indicated by the FIB. In Figure 2.2, the subscriber sends an INTEREST for the name `/aueb.gr/ai/new.htm` (arrows 1-3). If the PIT already contains an entry for the *exact* name, meaning that this *exact* content item had already been requested, the router adds the incoming interface to this PIT entry and discards the INTEREST, effectively forming a multicast tree for the content item. When a content item that matches the requested name is found at a publisher node or a CS, the INTEREST message is discarded and the content is returned in a DATA message. This message is forwarded back to subscriber(s) in a hop-by-hop manner, based on the state maintained in the PITs. Specifically, when a CR receives a DATA message, it first stores the corresponding content item in its CS and then it performs a longest-prefix match in its PIT to locate an entry matching the Data packet;[1] if a PIT entry lists multiple interfaces, the Data message is duplicated, thus achieving multicast delivery. Finally, the CR forwards the Data message packet to these interfaces and deletes the entry from the PIT (arrows 4-6). In case there are no matching entries in the PIT, the router discards the Data packet as a duplicate.

In NDN, security is built within the content. Each Data message contains a signature over the name and the content included in the message, plus information about the key used to produce the signature, e.g., the public key of the signer, a certificate for that public key or a pointer to them. This allows any node, to verify the binding between the (possibly, human-readable) name of the packet and the accompanying content. In order to verify that the content comes from an authorized source though, the subscriber must trust the owner of the public key used for signing. The hierarchical structure of names simplifies building trust relationships, for example, `/aueb.gr/ai/main.html` may be signed by the owner of the `/aueb.gr/ai` domain, whose key may be certified by the owner of the `/aueb.gr` domain.

2.3 PURSUIT

The *Publish Subscribe Internet Routing Paradigm* (PSIRP) [21] project and its continuation the *Publish Subscribe Internet Technology* (PURSUIT) [7] project, both funded by the EU Framework 7 Programme, have produced an architecture that completely replaces the IP protocol stack with a publish-subscribe protocol stack. The PURSUIT architecture consists of three separate functions: *rendezvous*, *topology management* and *forwarding*. When the rendezvous function matches a subscription to a publication, it directs the topology management function to create a route between the publisher and the subscriber. This route is finally used by the forwarding function to perform the actual transfer of data.

Information objects in PURSUIT are identified by a (statistically) unique pair of IDs, the *scope ID* and the *rendezvous ID*. The scope ID groups related con-

[1]The longest-prefix match is needed since the requested name may be a prefix of the one returned.

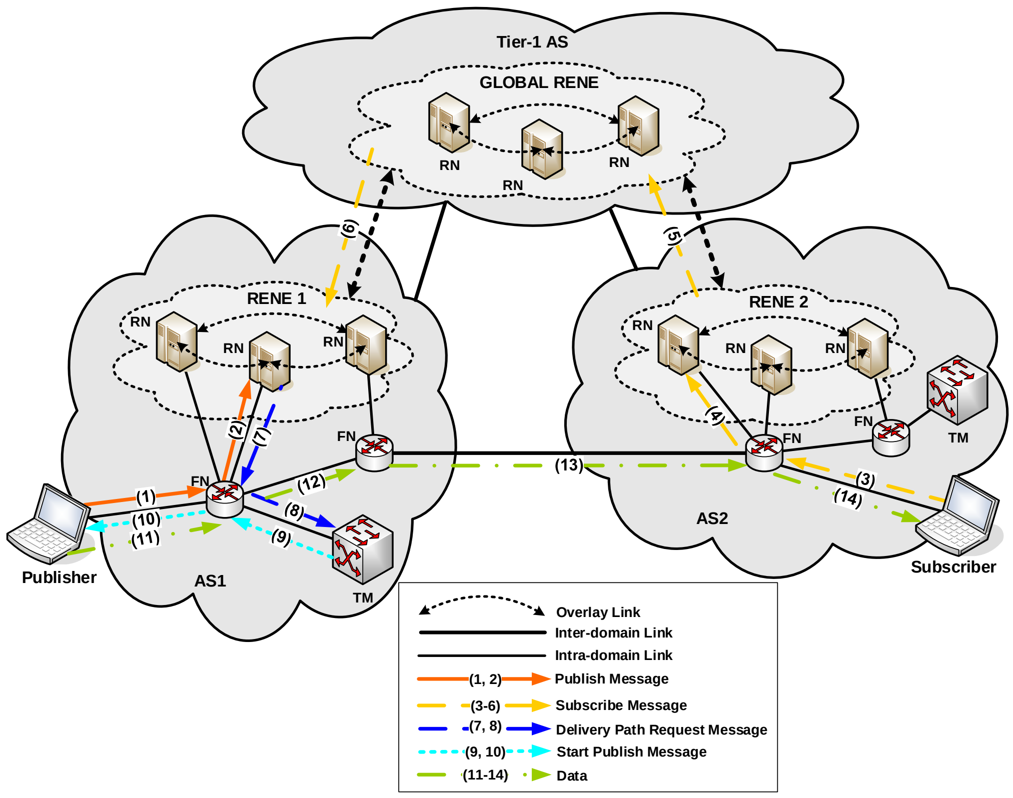


Figure 2.3: The PURSUIT architecture. RN stands for Rendezvous Node, RENE for REndezvous NEtwork, FN for Forwarding Node and TM for Topology Manager.

tent items while the rendezvous ID is the actual identity for a particular piece of content [22]. Information objects may belong to multiple scopes (possibly with different rendezvous IDs), but they must always belong to at least one scope. Scopes serve as a means of (a) defining sets of content items within a given context and (b) enforcing "boundaries" based on some dissemination strategy for the scope. For example, a publisher may place a photograph under a "friends" scope and a "family" scope, with each scope having different access rights. While PURSUIT names are flat as in DONA, scopes in PURSUIT can be organized in scope graphs of variable forms, including hierarchies, therefore a complete name consists of a sequence of scope IDs and a single rendezvous ID, thus generalizing the DONA naming scheme. Name resolution in PURSUIT is handled by the rendezvous function, which is implemented by a collection of *Rendezvous Nodes* (RNs), the *Rendezvous Network* (RENE), implemented as a hierarchical DHT [23, 24], as shown in Figure 2.3. When a publisher wants to advertise a content item, it issues a PUBLISH message to its local RN which is routed by the DHT to the RN assigned with the corresponding scope ID (arrows 1-2).[2] When a subscriber issues a SUBSCRIBE message for the same content item to its local RN, it is routed by the DHT to the same RN (arrows 3-6). The RN then instructs a *Topology Manager* (TM) node to create a route connecting the publisher with the subscriber for data delivery (arrows 7-8). The TM sends that route to the publisher in a START PUBLISH message (arrows 9-10), which finally uses this route to send the content item via a set of *Forwarding Nodes* (FNs). The actual delivery paths are calculated upon request by the rendezvous function as a series of links between FNs and encoded into source routes using a technique based on Bloom filters [25]. Specifically, each network node assigns a tag, i.e., a long bit string produced by a set of hash functions, to each of its outgoing links, and advertises these tags via the routing protocol. A path through the network is then encoded by ORing the tags of its constituent links and the resulting Bloom filter is included in each data packet. When a data packet arrives at a FN, the FN simply ANDs the tags of its outgoing links with the Bloom filter in the packet; if any tag matches, then the packet is forwarded

[2]Note that the RN assigned with a scope ID may reside outside the AS of its publisher, due to the way DHTs operate.

over the corresponding link [26]. In this manner, the only state maintained at the FNs is the link tags. Multicast transmission can be achieved by simply encoding the entire multicast tree into a single Bloom filter.

PURSUIT supports the *Packet Level Authentication* (PLA) technique [27] for encrypting and signing individual packets. This technique assures data integrity and confidentiality as well as malicious publisher accountability. PLA can be used to check packets either at FNs or at their final destination. The use of flat names also permits self-certifying names for immutable data objects, using the object's hash as the rendezvous ID. Moreover, paths encoded into Bloom filters can use dynamic link identifiers, making it impossible for an attacker to craft Bloom filters or even to reuse old Bloom filters to launch DoS attacks.

2.4 SAIL

The *Architecture and design for the future Internet* (4WARD) [28] project and its continuation *Scalable and Adaptive Internet Solutions* (SAIL) [8] , both funded by the EU Framework 7 Programme, are investigating designs for the Future Internet and ways to facilitate a smooth transition from the current Internet. The core of these two architectures is the so-called *Network of Information* (NetInf), which designs an ICN architecture that supports the exchange of named content items. Beyond the aspects covered below, the SAIL architecture includes many other services, such as searching for content items via keywords. The SAIL architecture is very general: it combines elements present in the NDN and PURSUIT approaches and can even operate in a hybrid mode. Furthermore, it can be implemented over different routing and forwarding technologies, by introducing *convergence layers* to translate SAIL messages to actual network packets [29].

Content item names in SAIL are "flat-ish": they provide some structure and they can even be hierarchical, but they do not carry location or organizational information. SAIL defines the `ni://A/L` URI scheme in which names consist of an authority part `A` and a local (with respect to the authority) part `L`. Each part can be a hash, thus allowing for self-certification, or any other type of string,

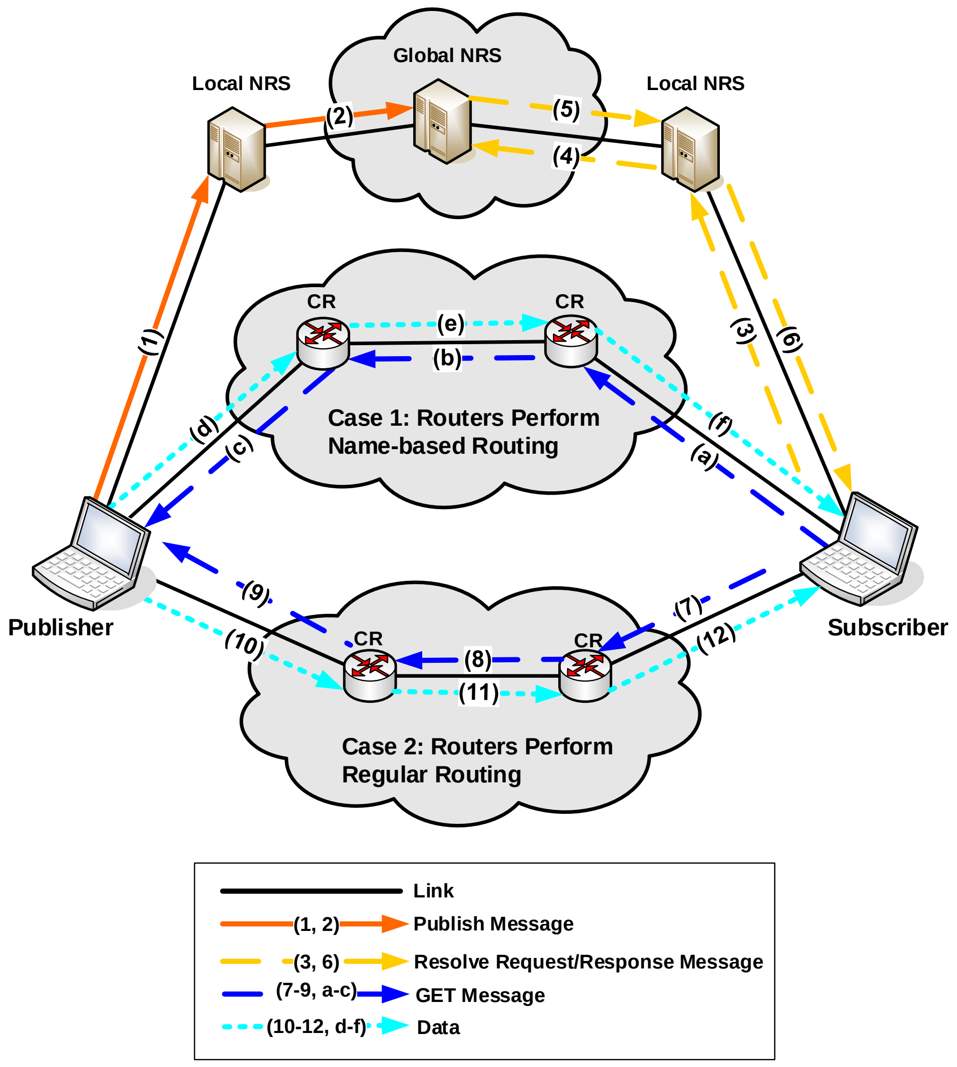


Figure 2.4: The SAIL architecture. NRS stands for Name Resolution System, CR for Content Router.

thus allowing for regular URLs [30]. SAIL names are considered flat for name comparison purposes, that is, a subscription will only match a publication if there is an exact name match between them, as in PURSUIT. On the other hand, SAIL names can be considered hierarchical when used for routing, that is, routers can use longest prefix matching to determine how to route a message, as in NDN [29].

A *Name Resolution System* (NRS) is used to map object names to locators that can be used to reach the corresponding content item, such as IP addresses (Figure 2.4). The NRS is some form of DHT, either a multilevel DHT [31] or a hierarchical SkipNet [32]. In the multilevel DHT solution, each authority maintains its own local NRS to handle the resolution of the `L` part, while a global NRS handles the resolution of the `A` part. A publisher makes a content item available by sending a Publish message with its locator to the local NRS, which stores the `L` to locator mapping (arrow 1). The local NRS aggregates all the `L` parts for the same authority `A` into a Bloom filter [25], and sends a Publish message to the global NRS (arrow 2). The global NRS stores the mapping between the authority `A` plus the Bloom filter and the local NRS, replacing any previous such mapping. When a subscriber is interested in a content item, it can send a Get message to the its local NRS which consults the global NRS (arrows 3-4) in order to return a locator for the object (arrows 4-5). Finally, the subscriber sends a Get message to the publisher, using the returned locator (arrows 7-9), and the publisher responds with the content item in a Data message (arrows 10-12).

The SAIL architecture envisions a fully-fledged security system that covers name security, content integrity, authentication and confidentiality, and authorization and provenance. The basic building block of the security architecture is the inclusion of hash values in names, which allows self-certification of both the authority and the local part. SAIL names may explicitly identify the hash scheme used, e.g., SHA-1, to allow many such schemes to co-exist [30].

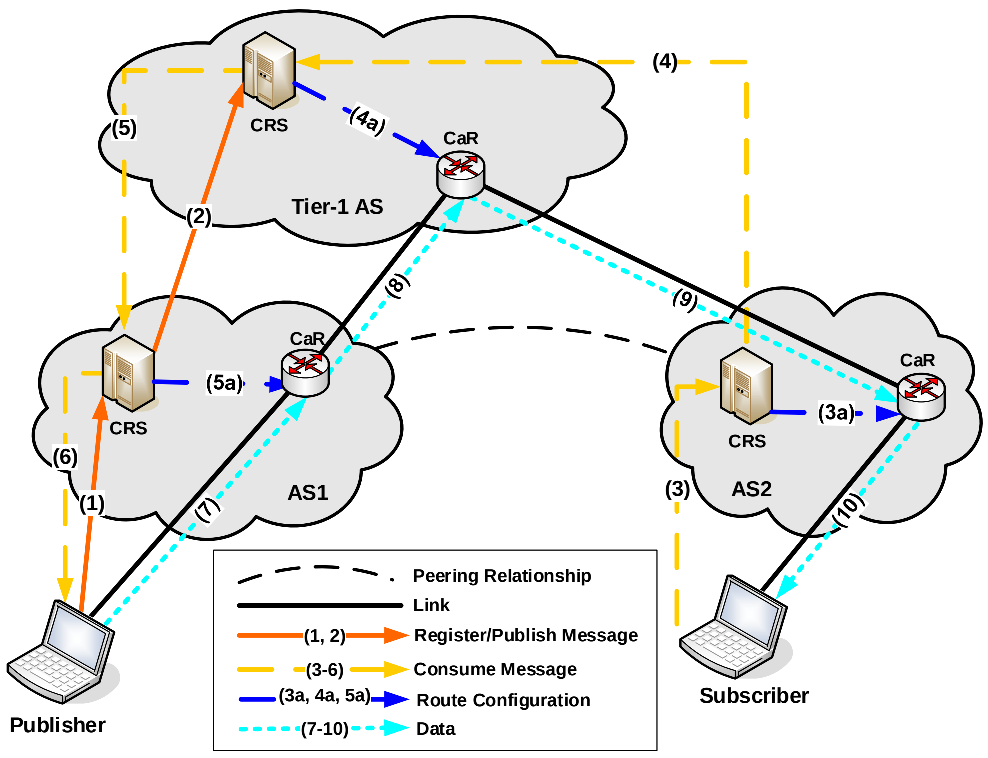


Figure 2.5: The coupled COMET architecture. CRS stands for Content Resolution System, CaR for Content-aware Router.

2.5 COMET

The *COntent Mediator architecture for content-aware nETworks* (COMET) [9] project, funded by the EU Framework 7 Programme, is designing mechanisms for optimizing content source selection and distribution by mapping content to appropriate hosts or servers based on transmission requirements, user preferences, and network state [33]. The core component of the COMET architecture is a *Content Mediation Plane* (CMP) which mediates between the network providers and the content servers, being aware of both content and infrastructure. The COMET project has produced two very different architectures for the CMP: a coupled design called *Content-Ubiquitous Resolution and Delivery Infrastructure for Next Generation Services* (CURLING) [34], which is an ICN architecture with coupled name resolution and routing, and a decoupled design that enhances content delivery without fundamentally changing the underlying Internet [35]. Unlike other ICN approaches which strive for location independence, COMET allows both subscribers and publishers to explicitly include location preferences for content, following established business practices. For example, a subscriber may ask for bookstores in a specific country, and a publisher may only make videos available to a specific country.

A precise naming scheme has not been defined for COMET. However, in COMET the content names are provided by a *Content Resolution System* (CRS) when the content is registered by the publishers, thus allowing names for related content to be explicitly aggregatable, e.g., episodes of a TV series can have sequential names. This allows the naming system to scale by exploiting existing relationships between content items. COMET adopt two approaches for locating and disseminating content. The first apprach is presented in Figure 2.5. A publisher that wants to make some content available sends a REGISTER message to its local CRS node which issues a name for the content and stores the actual location of the content, e.g., the IP address of the publisher (arrow 1). This information is propagated upstream in the AS hierarchy using PUBLISH messages, so that each parent CRS ends up with a pointer to its child CRS that sent the PUBLISH message

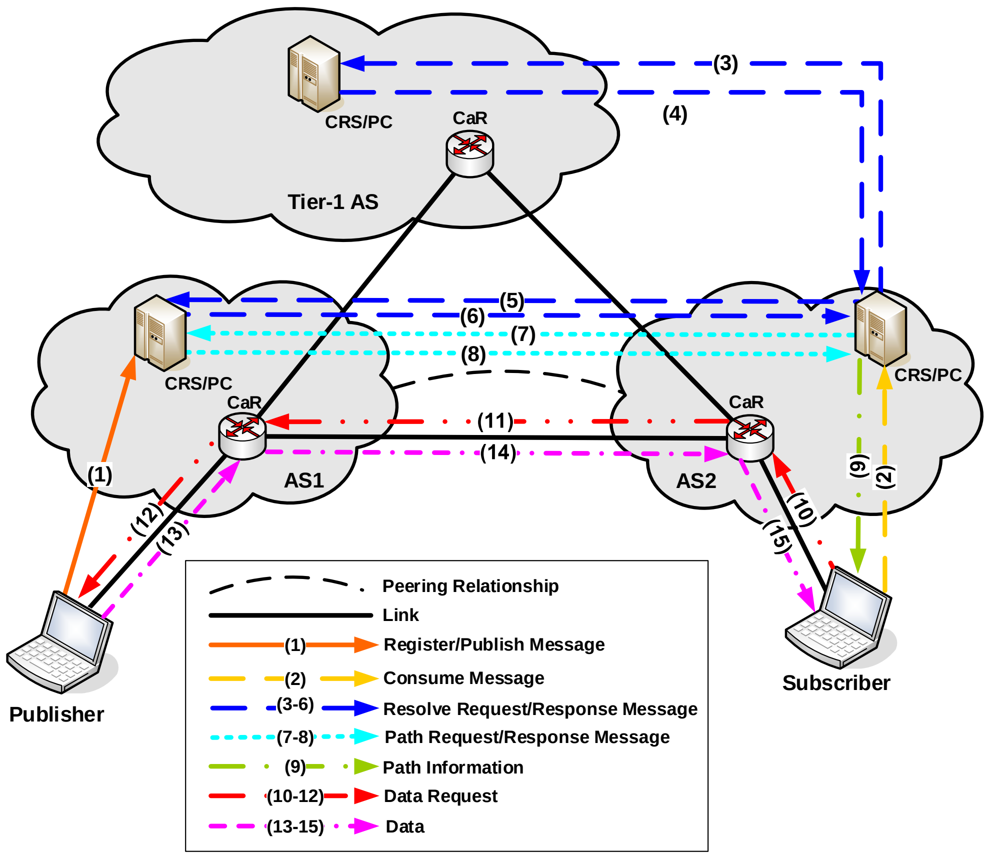


Figure 2.6: The decoupled COMET architecture. CRS stands for Content Resolution System, CaR for Content-aware Router, PC for Path Configurator.

(arrow 2). The publisher may limit the propagation of this information to a specific area, e.g., an IP prefix, so Publish messages may not reach the Tier-1 provider. A subscriber that is interested in some content issues a Consume message to its local CRS, which is similarly propagated upwards in the CRS hierarchy until it reaches a CRS that has information about that name (arrows 3-4). The subscriber may either limit the propagation of this information to a specific area or exclude specific areas from this propagation. When a match is found, the Consume message follows the pointers in the CRSs to reach the actual publisher (arrows 5-6). As the Consume message travels from the subscriber to the publisher, each CRS on the way installs forwarding state at the *Content-aware Routers* (CaRs) of each intermediate AS, pointing back towards the subscriber (arrows 3a-5a). The publisher can thus send the corresponding data to the subscriber by using these pointers (arrows 7-10).

The second approach is presented in Figure 2.6. In this case, the CRS system is similar to DNS, in that the CRSs split the object namespace among themselves in a fixed hierarchical manner. This means that when a publisher wants to make some content available, it simply sends a Register message to its local CRS (arrow 1), which is not propagated further because it must belong to the namespace assigned to that CRS. When a subscriber issues a Consume message for some content (arrow 2), this is resolved by the root CRS to a pointer towards the publisher's CRS (arrows 3-4). The subscriber's CRS contacts the publisher's CRS to get the location of the publisher (arrows 5-6), e.g., its IP address. Then the subscriber's *Path Configurator* (PC) contacts the publisher's PC (shown co-located with the CRS nodes for simplicity) requesting a source route from the subscriber to the publisher (arrows 7-8). This source route is returned to the subscriber (arrow 9) which uses it to request content (arrows 10-12); its reverse is used by the publisher to return the content (arrows 13-15).

COMET adopts security techniques from other ICN architectures [35]. The security techniques that may be used depend however on the exact naming structure used. For example, if related pieces of content use sequential names for aggregation purposes, these names cannot use the self-certification approach of DONA which relies on embedding hashes in the names.

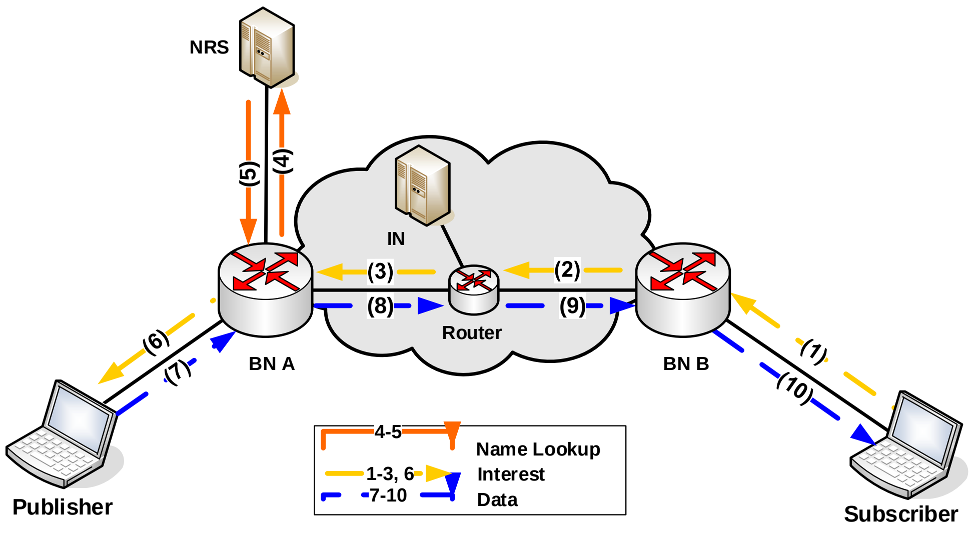


Figure 2.7: The CONVERGENCE architecture. NRS stands for Name Resolution System, BN for Border Node, IN for Internal Node.

2.6 CONVERGENCE

The CONVERGENCE [10] project, funded by the EU Framework 7 Programme, envisions an ICN-based Future Internet that facilitates user access to content, spanning from digital data and services to people and real-world objects. Each such object in CONVERGENCE is represented by a *Versatile Digital Item* (VDI), a common container for all kinds of digital content, based on the MPEG-21 specification. A *Content Network* (CONET) [36] allows publishers to make available VDIs and subscribers to express interest in those VDIs. A distinguishing characteristic of the CONVERGENCE architecture is that it attempts to ease transition from IP by reusing existing functionality. For example, since CONVERGENCE messages are expected to be large due to naming and security meta-data, rules are defined for splitting them to *carrier packets*, e.g., IP datagrams. Furthermore, an IP header option has been defined to carry the essential information from CONVERGENCE message headers, allowing CONVERGENCE-aware IP routers to treat IP datagrams containing CONVERGENCE messages differently. In CONVERGENCE object names consist of a `namespace ID` and a `name` part, whose format is determined by the `namespace ID`. While the default format of CONVERGENCE names is similar to that in DONA, i.e., a flat `P:L` pair [36], hierarchical names may also be used as in NDN [37], or even URLs. The exact properties of the names depend therefore on the specific namespace used. Since CONVERGENCE is most similar to NDN, we assume in the following the use of hierarchical names.

The CONVERGENCE architecture, shown in Figure 2.7, has many similarities with NDN; indeed, its prototype has been implemented as a modification of the NDN prototype [37]. Subscribers issue INTEREST messages requesting a content item, which are forwarded hop-by-hop by *Border Nodes* (BNs) to publishers or *Internal Nodes* (INs) that provide caching (arrows 1-3 and 6). Publishers respond with DATA messages which follow the reverse path (arrows 7-10). In order to reduce the state requirements at the BNs, CONET diverges from NDN in three aspects. First, BNs do not maintain name-based routing information for every advertised name prefix, but only for a small portion of them, hence their routing table operates like a route cache. If an INTEREST message cannot be forwarded because there is no routing information for the corresponding name, the BN consults an external *Name Resolution System* (NRS), e.g., DNS, in order to find out how to forward the INTEREST (arrows 4-5). Second, as INTEREST messages are propagated they accumulate the network addresses of the BNs they pass, allowing the publisher to route the DATA message by reversing this path information, without requiring the maintenance of pointers at BNs. Third, BNs do not have to be directly connected; instead, the path between two BNs can involve multiple hops, e.g., via IP routers as shown in Figure 2.7, hence their designation as *border* nodes. Therefore, unlike CRs in NDN, BNs map names to network addresses, e.g., IP addresses, rather than to interfaces.

CONVERGENCE adopts the per DATA message security approach of NDN, i.e., each DATA message contains a digital signature. Due to the large overhead of the meta-data required for signature verification, DATA messages are expected to be much larger than carrier packets, e.g., IP datagrams encapsulated in Ethernet frames. For this reason, CONVERGENCE proposes performing security checks on content only at the DATA message level at the subscriber.

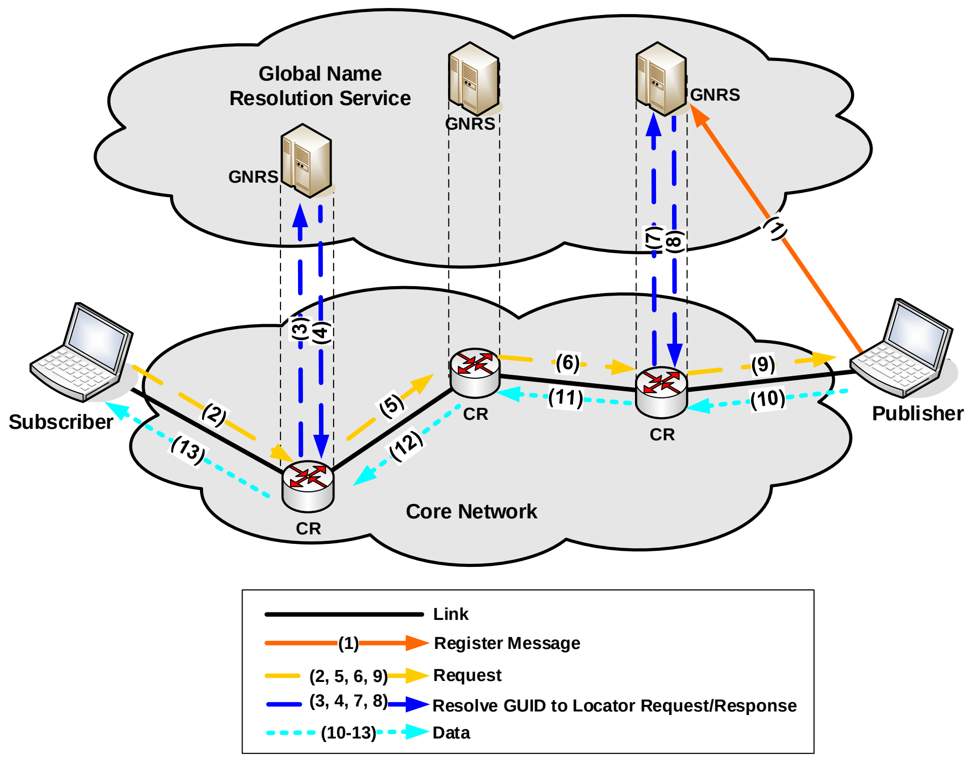


Figure 2.8: The MobilityFirst architecture. GNRS stands for Global Name Resolution Service, CR for Content Router.

2.7 MobilityFirst

The *MobilityFirst* [11] project, funded by the US Future Internet Architecture program, proposes a clean-slate Future Internet architecture with an emphasis on treating mobile devices as first-class citizens [38]. As a result, MobilityFirst provides detailed mechanisms to handle both mobility and wireless links, as well as multicast, multi-homing, in-network caching and security. The basis of the MobilityFirst architecture is the separation of names for *all* entities attached to the network (including content items, devices and services) from their network addresses: each entity has a globally unique name, which can be translated into one or more network addresses at various points in the network, thus allowing messages to be dynamically redirected in order to follow a mobile device or content.

Each network entity in MobilityFirst is assigned a *Globally Unique Identifier* (GUID) via a global naming service that translates human-readable names to GUIDs. Every device in MobilityFirst must obtain GUIDs for itself, its content items, and its services. GUIDs are flat 160-bit strings with no semantic structure and they may be randomly selected, since their length ensures that the probability of a collision is small. Alternatively, GUIDs can be self-certifying hashes of content items, thus allowing content integrity verification, or hashes of public keys, thus binding devices to principals. Each network attached entity has a unique GUID, and if an entity (e.g., video file) is available in multiple network locations, then all of its copies will have the same GUID. By naming all network entities, MobilityFirst can support both name-based content delivery (via content GUIDs) and host-to-host communication (via device GUIDs).

In MobilityFirst all communication starts with GUIDs, which are translated to network addresses in one or more steps, via a *Global Name Resolution Service* (GNRS) as shown in Figure 2.8. A publisher that wishes to make some content available asks the naming service for a GUID and then registers the GUID with its network address in the GNRS (arrow 1). A GUID is mapped via hashing to a set of GNRS server addresses, which are contacted using regular routing [39]. When a subscriber wants to receive some content, it sends a GET message that includes the GUID of the requested object, along with its own GUID for the response, to its local *Content Router* (CR) (arrow 2). The CR can only route based on actual network addresses, e.g., IP addresses, hence it asks the GNRS for a mapping between the destination GUID and one or more network addresses (arrow 3). The GNRS replies (arrow 4) with a set of network addresses (optionally it may also send a source route, a partial source route and/or intermediate network addresses). The CR selects one of these network addresses, adds it to the GET message, which it then forwards using the regular routing tables in the CRs (arrows 5-6 and 9). The GET message includes both the destination GUID and the destination network address, and any CR along the path can consult the GNRS to receive an updated list of network addresses for the destination GUID (arrows 7-8) if, for example, due to mobility the GET message cannot be delivered to the publisher. The publisher sends its response to the subscriber's GUID, using the same procedure (arrows 10-13).

MobilityFirst envisions a decentralized trust model for name certification, where independent naming organizations exist (e.g., one per country or one per institute) for mapping human-readable names to GUIDs. The GUID of an entity can be securely bound to that entity via cryptographic techniques, thus enabling traffic accountability. On the other hand, users can frequently request a new GUID to avert profiling.

2.8 A common ICN model

In this section we define an architectural model that we use as a reference in the reminder of this dissertation. We refer to this model as the Publish-Subscribe Internetworking (PSI) architecture. PSI is not mapped to a particular ICN architecture, but rather tries to capture the commonalities of the previously discussed architectures. In this section we define the *roles* and the *functions* from which PSI is composed, leaving the particular *implementation choices* abstract.

PSI is composed of the following entities:

- Information Owner (or simply Owner): The entity that creates and owns a content item. The owner is responsible for assigning names to content items and for creating (if necessary) access control rules that govern who can access each item. The role of owner captures real world entities (e.g., an author, a university, a company, a government).

- Publisher: A network entity that actually hosts a content item. A publisher may be under the full control of an owner (e.g., the Web server of a university), but it may also be (semi-)independent (e.g., proxy caches and CDN servers). Publishers may either have been appointed by the owners themselves (e.g., a university may host a content item in its Web server, or pay a CDN to host it), or may act opportunistically (e.g., an in-network cache).

- Subscriber: A network entity used by a real world entity that is interested in receiving a content item (e.g., a PC, a mobile phone).
- Rendezvous Node (RN): A network entity that acts as an indirection point between subscribers and publishers. A RN's main functionality is to accommodate subscriber interests for particular content items. All the RNs of an architecture form the *rendezvous network* (RENE).

These entities interact with each other in the following manner: An owner creates a content item, assigns a *name* and appoints (at least) one publisher to host that item. Each content item is uniquely identified by a hierarchical *identifier*. Publishers *advertise* the content items they host. The advertisement of an item is received and stored by a RN in the RENE. A subscriber sends a content *subscription* request that is routed through the RENE and using a *resolution* process reaches the RN that has a matching entry for the item of interest. A subscription request may explicitly contain a content identifier or it may imply it (e.g., a subscriber may send a subscription request for "Finance/Stocks/Top") A successful match will ultimately result in the content being *forwarded* from a publisher to the interested subscriber(s). The RENE is (conceptually) organized as a directed acyclic graph (DAG) with every node of the graph being a RN and each edge of the graph being a pointer to another RN. Each RN is responsible for managing certain portions of the identifier space and the identifier of a content item denotes the path to the RN that manages it. In the example of Figure 2.9, all item identifiers with prefix $Travel/Greece/Hotels/$ will be managed by the bottom-right RN. Moreover, as it can be observed from this figure, many RNs at the same level can manage the same (partial) identifier (in our example there are two RNs at the second level managing the partial identifier "Greece"); the RN that will be used during the resolution process is determined by the complete identifier (e.g., the leftmost RN will be used for identifiers with prefix $Travel/Greece$ whereas the rightmost RN will be used for identifiers with prefix *Education*/*Greece*)

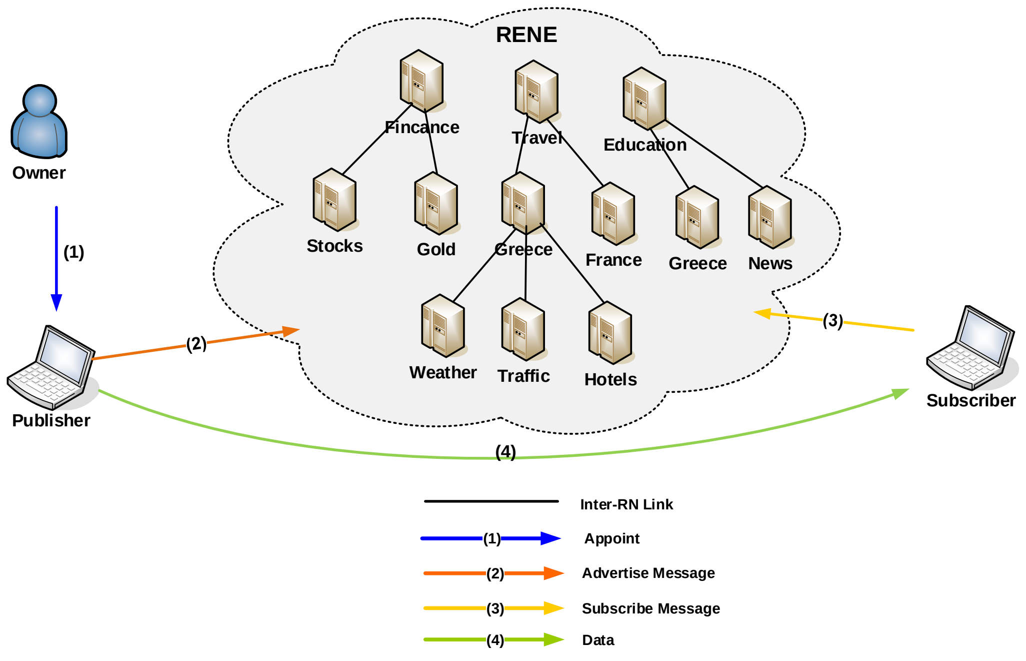


Figure 2.9: The PSI architecture.

Chapter 3

ICN security requirements and existing solutions

3.1 Content-related requirements and solutions

3.1.1 Access control

The need for access control in ICN is stronger than in current IP networks. The reason is that in ICN, content items do not always lie in the administrative realm of their owner due to caching and content replication (which is facilitated by ICN). It must be possible therefore to enforce access control policies for content items scattered around the network. This requirement raises some new challenges, such as enabling caches and content replication points to enforce access control policies without compromising subscriber privacy, as well as, without revealing subscriber credentials, or even the access control policy itself.

Zhu et al. [40] developed a scheme that supports access control for an ICN-based audio conference tool. In their approach, a user is chosen as the "organizer" of a conference call. The organizer creates a pair of keys: an encryption key and a decryption key. Then he delivers the decryption key to each participant of the conference, by encrypting it with the participant's public key. Finally he encrypts all the information required in order to establish the audio conference, using the generated encryption key; only authorized users that have the decryption key can decrypt the transmitted information. Jacobson et al. [41] introduced a custodian-based sharing scheme. Custodians are entities that have an explicit relationship with a collection of data and can be accessed through many (prioritized) endpoints. Custodians not only can set access control policies to the content items they host, but they can also set rules regarding the conditions under which an endpoint can be accessed (e.g., an endpoint that resides in a smartphone can be set to be accessibly only when the smartphone is connected to a Wi-Fi network).

3.1.2 Content confidentiality and integrity

A common security requirement for ICN is the protection of content *confidentiality* and *integrity*. Content confidentiality assures that a piece of content can be accessed only by its intended recipients. Content integrity guarantees that a piece of content has not been modified during its transmission. In the current Internet, confidentiality and integrity are usually provided by mechanisms that are based on a network of trusted third parties. As an example every operating system is pre-configured with the public keys of trusted entities, which form the pillars of a global network of trust. However this model has many weaknesses and a departure from it is highly desirable [42]. Content confidentiality and integrity mechanisms in ICN should rely on the content identifier, rather than on location-based or publisher-based certificates. This property significantly facilitates content replication, caching and multihoming. Moreover a content confidentiality and integrity mechanism should not prohibit the usage of mutable and/or human readable content identities.

Zhang et al. [43] utilized content names and Identity Based Encryption (IBE) in order to provide content confidentiality and integrity. IBE is an asymmetric encryption scheme, which allows the encryption of plaintext using an identity and some known public system parameters. In their work Zhang et al. use IBE to enable the exchange of symmetric encryption keys, which are then used for encrypting exchanged content. Moreover, each owner digitally signs his content items using the private key that corresponds to his identity, therefore content integrity is also protected. Koponen et al. [5] assure content integrity in their ICN architecture by digitally signing every content item and by including the public part of the key used for the signature generation in the content identifier. More precisely, every content "principal" (i.e., the term used for owners in their architecture) identified by a public key P, generates a label L for each content item he owns. The identifier of an item then becomes $Hash(P)||L$, where $Hash(P)$ is the cryptographic hash of the public key P and $||$ denotes concatenation; such names are a form of self-certifying names. All forwarded items include P appended to their data, as well as a digital signature over the content data, generated with the private counterpart of P; using this digital signature and P, subscribers are able to verify the integrity of the received data. Smetters et al. [20] proposed another solution that assures content integrity based on content identifiers. Their solution allows the usage of human readable names and operates as follows: a content owner generates a name N of a piece of content and calculates its hash H, then a publisher advertises the triple $(N, H, Sign(N||H))$, where $Sign(N||H)$ is a digital signature of the concatenation of N and H using the owner's private key. A subscriber can send a subscription message for a piece of content using its name N. Upon receiving the content and in order to verify its integrity, the subscriber calculates the hash H and verifies the digital signature $Sign(N, H)$.

3.1.3 Content provenance verification and authentication

Another important security requirement from ICN is content provenance verification and content authentication. Content provenance verification refers to the verification the owner of a piece of content. In a secure ICN architecture, it should be impossible for a (malicious) user to impersonate another user, and also impossible for him to manipulate a piece of content in such a way that he appears to be the owner of that piece of content. Proper content provenance mechanisms enable the deployment of effective accountability solutions and prevent man-in-the-middle attacks. Content authentication refers to the binding of a content item name to the item data. Every piece of content is generally composed of two parts: its name and its data (e.g., consider a movie encoded in an mpeg-2 file named "Movie X.avi," the name of this piece of content is "Movie X.avi," whereas its data are the bytes that form the mpeg-2 file). In an ICN architecture it should be possible to verify the binding of these two parts, i.e., given a content name and some data, it should be possible to verify if they belong to the same content item or not. Content authentication is orthogonal to content integrity: if a subscriber subscribes for “Movie X” and receives “Virus A,” the content integrity check will simply verify that “Virus A” has not been modified during transmission, whereas the authentication check will reveal that the content item the subscriber received is not what it requested. Content authentication is an essential requirement for ICN architectures, in order to filter out malicious content and prevent attacks such as phishing and spamming.

Wong and Nikander [44] proposed a solution that uses a triple identifier for each piece of content: the authority identifier, the content identifier and the location identifier. In order for a subscriber to send a subscription message for an item, it has to be aware of the authority (i.e., owner) identifier as well as the content identifier. Initially it uses a (reliable) directory service in order to map an authority identifier to secure meta-data–such as the public key of the authority. Then it uses a (reliable) resolution service in order to map the (authority identifier/content identifier) pair to a location identifier. In addition to the location identifier the resolution service provides meta-data about the content item–such as its hash. The meta-data received from the directory service is also securely embedded in the received content; therefore its provenance and authenticity can be verified. Ghodsi et al. [45] proposed a solution in which content items are identified using self-certifying identifiers of the form $Hash(P) : L$, where Hash(P) is the cryptographic hash of the content owner’s public key P, and L is a label chosen by the content owner. A digital signature embedded in the content, allows a subscriber to verify content provenance. Note, however, that such an approach requires a reliable resolution service to map human readable names to Ps.

It should be noted here that all the solutions proposed for content integrity can also be used for content provenance verification, since they are based on the public key of the owner. The solution developed by Koponen et al. [5] includes in the content body the public key of the owner, as well as, a digital signature generated using the owner's private key, therefore content provenance can be verified, simply by validating the digital signature. The solution developed by Smetters et al. [20] can also be used for content provenance verification, since the public key of the content owner is included in the content header, and the content name and its hash are digitally signed with the content owner's private key. The IBE based scheme proposed by Zhang et al. [43] enables content provenance verification, since content is digitally signed by the private key of the owner; IBE enables any entity to verify the binding between the signature and the identity of the owner of the content.

3.2 Publisher/Subscriber-related requirements and solutions

3.2.1 Endpoints privacy

In some application scenarios it may be desirable to preserve *publisher privacy*. Publisher privacy can be potentially violated through information revealed by an advertisement message or by a forwarded item. Securing advertisement messages may imply hiding the identity of a publisher from the RENE. However, hiding publisher identity makes it hard to hold misbehaving publishers accountable, thus, facilitating the launch of attacks such as phishing and spamming. On the other hand, protecting publisher privacy by securing forwarded items is a more reasonable and feasible option: the only publisher-specific information that may be required in a packet carrying content to a subscriber is the identifier of a path leading back to the publisher, in the case where closed-loop transport control is required. PURSUIT's Bloom filter based forwarding [26] contributes to publisher privacy preservation: by observing a Bloom filter, it is not easy to determine the origin or the destination of a packet[1], therefore publisher (location) privacy is preserved. When crumb-based forwarding is used, as in NDN [46], forwarded

[1]A Bloom filter is composed by ORing all the identifiers of the links that a packet has to traverse: the output of the OR operation does not give any information about the first link of the path.

items contain only the content identifier; if the content identifier does not reveal information about the publisher, then publisher privacy is protected.

In contrast to publisher privacy, preserving subscriber privacy is a more challenging requirement. A characteristic of ICN architectures that affects subscriber privacy is that the content identifier that may be included in a subscription message reveals more information about subscriber preferences, compared to an IP header in a connection request in a legacy IP network. Moreover, the identifier of the content being requested is available to all RNs that process a request, making privacy an even more challenging problem [47]. What is worse, a malicious user that is able to "overhear" a subscription message, can easily learn the exact content in which the "victim" is interested in, simply by subscribing to the same content (if no access control mechanisms are used). On the other hand, unlike IP networks where a packet includes the source IP address, a subscription message (in some ICN architectures–such as NDN) does not contain the identity of the requesting entity.

Caching, which is a key functionality provided by ICN, can also be a threat to subscriber privacy. Lauinger et al. [48], used response times in order to estimate if a piece of content is cached close to the attacker: small response times means that the requested content is cached close to the attacker. The fact that a piece of content is cached close to the attacker is an indication that another user, in the vicinity of the attacker, has requested the same content. Finally, another aspect of subscriber privacy that is receiving increasing attention is that of Decisional Interference [49], i.e., the ability of an attacker to filter (censor) the information that a subscriber can access.

Subscriber privacy preservation usually refers to one (or more) of the following sub-requirements: *anonymity*, *unlinkability*, and *unobservability* [50]. Anonymity is the ability to hide a user identity within a set (the anonymity set), i.e., every subscriber should not be identifiable within the anonymity set. The anonymity set is the finest grained information a malicious entity can learn about a subscriber. The bigger the anonymity set is, the better the privacy for the user. Unlinkability of two (or more) items, means that by observing these items it is not possible to learn any information about their relationship. In the context of ICN this means that by observing messages (e.g., subscriptions and advertisements) it should not be possible to correlate these messages to each other, nor to associate these messages to a subscriber (or publisher) identity. Unobservability means that the object requested by a particular user cannot be distinguished from a set of objects, i.e., given a subscriber identity, a subscription message that originated from that subscriber and a set of content items, it should be not possible to determine the specific item from the set that is requested with the subscription. Unobservability and anonymity are two different properties: when unobservability is used the identity of the subscriber is not necessarily hidden.

PURSUIT's Bloom filter based forwarding based forwarding [26] contributes to subscriber's anonymity and unlikanbility: by observing a Bloom filter it is not possible to determine the (location) identity of the subscriber, and it is not possible to link a forwarded item to a known subscription or advertisement message. DiBenedetto et al. [51] proposed a Tor-like anonymization network for the NDN ICN architecture, code-named ANDaNA, which provides anonymity and unlikability. In ANDaNA, before sending a content request, a subscriber selects two "anonymizing routers," the entry router and the exit router, and distributes different symmetric encryption keys to each of them. The subscriber encrypts his subscription using the public keys of the routers, and sends it to the entry router, which then forwards it to the exit router. When the exit router receives the corresponding publication, it encrypts it using the symmetric key that has received by the subscriber and forwards it to the entry router. Then the entry router encrypts once more the received ciphertext with its own symmetric key (provided by the subscriber) and forwards it to the subscriber. Finally, the subscriber decrypts the response. Hsiao et al. [52] proposed a similar system which has better performance, by relaxing the threat model: this work assumes that the autonomous system where the subscriber is located is reliable and resilient to attackers. Based on this assumption the entry point to the anonymizing network is always a (trusted) router located inside the subscriber's autonomous system. Arianfar et al. [53] proposed a solution that offers unobservability to subscribers. In their approach, a publisher splits the file he wants to protect in n blocks, t_1, t_2, ..., t_n, and creates a "cover file" which is also composed of n blocks (c_1, c_2, ..., c_n). All file blocks and the corresponding cover file blocks are assumed to have the same length. The publisher then, applies a reversible randomizing function $r()$ to every block and advertises all the (randomized) blocks of the cover file (i.e., $r(c_1)$, $r(c_2)$, ..., $r(c_n)$)) as well as chunks that are created by XORing a (randomized) file block with a (randomized) covered file block (e.g., $r(t_1)$ XOR $r(c_2)$, $r(t_3)$ XOR $r(c_1)$)). In order for a subscriber to receive a file block she has to send a subscription message for the appropriate cover file blocks and chucks (e.g., in order to receive t_1 she must send a subscription message for $r(c_2)$ and for $r(t_1)$ XOR $r(c_2)$; then she will be able to compose t_1, simply by XORing the received packets). The identifier used for the i^{th} published block of the cover file c is $H(H(c)||i)$, where H is a well known function and || denotes concatenation. The name used for a published chunk, composed of XORing the k^{th} block of the cover file with the l^{th} file block of a file t is $H(H(c)||k||H(t)||l)$. A subscriber learns H, n through a secure channel therefore she is able to send subscription messages for any combination of files.

3.3 Infrastructure-related requirements and solutions

3.3.1 RENE security and availability

The rendezvous functionality is probably the most important component of an ICN architecture. The rendezvous functionality should be designed with trust in mind, it should be fault tolerant, it should provide resilience to failures, and it should provide mechanisms that promote the cooperation between reliable nodes while isolating the unreliable ones [54]. Moreover, the rendezvous network should be able to cope with content subscription and content advertisement Denial of Service (DoS) attacks [55]. Content subscription DoS attacks are conducted by flooding a particular rendezvous node with subscription requests for the same content item, using many dispersed attacking entities (bots). Content advertisement DoS attacks on the other hand are conducted by advertising many different content items. Moreover, RENE may (deliberately or not) contribute to a DoS attack against subscribers, by not filtering the distribution of fake content. This phenomenon can be amplified in the case of poisoned caches [56]. Rendezvous availability may also be affected by the way it handles subscriptions to non-existent items: if a RN stores subscriptions for non-existent items, its resources may be saturated by an attacker that performs subscriptions for many items that do not exist [57].

Distributed Hash Tables (DHTs) are a common design choice for building (overlay) RENEs, since they provide load balancing and flat identifier resolution. Moreover DHTs are more resistant to failures–compared to a hierarchical structure–and allow for better governance of the information space–since all nodes are equally powerful. On the other hand DTHs introduce latency as an identifier resolution does not follow the optimal underlay network path. In Katsaros et al. [24] we proposed a hierarchical DHT scheme that can be used for inter-domain identifier resolution in ICN. The scheme–named HPastry–is based on the adaptation of the Pastry DHT [58] in order to support "route locality" and "route convergence" properties. The route locality property assures that an overlay route from one node to another node, both belonging to the same domain, will never contain links outside their domain. The route convergence property assures that two overlay routes originating from two different nodes in a domain, both targeting a node which is located outside their domain, will always exit their domain through the same node. This property is crucial for supporting efficient caching services. HPastry tries to take advantage of existing peering agreements and to avoid routing policies violations. D'Ambrosio et al. [31] have developed a multi-level DHT that can also be used as an identifier resolution service for ICN architectures. In this scheme various layers of DHTs–called areas– are considered. Areas can be mapped to the network physical topology (e.g., each AS can have its own area, and all AS-level areas can form a global DHT). An identifier resolution process starts from the node's local area; if the identifier is not found, then the resolution continues in a higher layer area. This process is repeated until a record for that identifier is found.

In contrast to a DHT-based, a hierarchical based RENE offers better resolution speed, as it does not introduce any additional stretch. However since the load is not equally distributed to all RNs, the availability of such a RENE may by jeopardized even by a limited set of (top level) RNs. Vasilakos et al. [59] explored the impact of the state that the DONA architecture would introduce in a hierarchical RENE, using a realistic network graph, and they show inequalities when it comes to load distribution. In order to overcome these inequalities they leveraged the role of Cloud computing and they show that with a cost at the resolution speed, some resolution requests can be outsourced to Cloud networks, therefore the availability of the overloaded nodes is significantly increased. Moreover they found that DONA can use Cloud systems in order to increase the amount of the information it can cache. Finally another factor that affects RENE availability is the speed of subscription/advertisement matching. This matching takes place over identifiers of unlimited length and it may have to consider additional parameters specified by a subscriber, e.g., a subscriber may have requested that matching should return the latest version of a content item. Yuan et al. [60] proposed some optimizations in the context of NDN architecture that improve significantly the matching speed. They proposed a matching algorithm optimized for specific identifier length and a compression algorithm for shortening longer names. Moreover they proposed simpler data structures, new packet encoding/decoding schemes, as well as, differentiation on the features that each RN should offer depending on its network position: highly loaded nodes should support fewer features.

3.3.2 Forwarding plane security and availability

The availability of the forwarding plane is of significant importance. The forwarding plane should be resistant to various attacks, which may lead to service interruption and denial of service. Sarela et al. [61] have identified three anomalies that can be caused in Bloom-filter based forwarding planes due to false positives, namely: packet storms, forwarding loops, and flow duplication. Packet storms can be caused in the core network, where each forwarding node is connected to many others; in these dense network areas each false positive will likely cause another false positive, leading to a packet storm. A forwarding loop can arise from a single false positive packet which is sent back to a node that has already traversed, or from a packet the Bloom-filter of which has been manipulated by an attacker in order to cause this loop. In some cases a packet can be re-transmitted to a subtree of the network topology even if the topology is loop free. This anomaly can be caused even in valley-free forwarding paths or even if small TTL is used for the packets. These anomalies can be surmounted by varying Bloom-filter parameters and by using bit permutations. The first part of the solution concerns the maximum number of bits that can be set to 1 in a link identifier, as well as the length of the Bloom-filter. The number of bits set to 1 should vary for each forwarding node, depending on its number of connections: highly connected nodes should be allowed to use more 1s in their link identifiers. Moreover the length of the Bloom-filter, should be variable: initially when a delivery tree is calculated a large Bloom-filter is created, then this Bloom-filter is "folded," by ORing the intersected values; the filter can be folded more than once, as long as the "folded" filter maintains a small estimated false positive ratio. The second part of the solution concerns how a Bloom-filter is constructed. In its original form, a Bloom-filter was constructed by ORing the link identifiers that form the delivery tree. Sarela et al. proposed an alternative construction method: every time a link is added to the Bloom-filter (using OR) the filter is permuted; the same permutation takes place during forwarding as well, i.e., every time a forwarding node forwards a packet, it permutes the bits found in the Bloom-filter of the packet header. Due to this permutation, if a packet returns to a node which it already has traversed, it will have a different Bloom-filter, therefore it will not be forwarded to the same links, avoiding this way loops and flow duplication. Forwarding plane availability can be improved also using caching, as caches decrease the network load

Forwarding availability may as well be affected by the in-network content verification mechanisms: an attacker may send subscription messages for a large amount of content items belonging to different owners–therefore protected with different security keys–saturating the resources of the forwarding nodes by making them verifying unnecessary data [57]. Content verification combined with bad transport decisions may also affect the availability of the forwarding plane [37]: if content is transmitted in small chunks, then forwarding nodes will be kept busy by performing many verifications, on the other hand if big chunks are used they will be fragmented in order to fit the layer 2 MTU. If a single fragment is lost then the verification of the chunk will fail, and the whole chunk will be re-requested. Salsano et al. [37] proposed an information-centric transport protocol that mitigates this problem. The proposed protocol splits a (large) chunk into fragments that fit the MTU of the layer 2 technology. Chunk verification tests are performed by nodes that have all the fragments of a chunk; if a verification fails due to a missing fragment, this specific fragment is re-transmitted, not the whole chunk.

Finally forwarding plane availability is greatly supported by the caching system, therefore attacks on caches may degrade forwarding plane availability. Chai et al. [62] have shown that a new caching model shall be adopted by ICN architectures. This model leverages probabilistic caching and achieves high cache hit ratios by limiting the number of caches in which an object is cached.

3.3.3 Application layer security and availability

Spamming is a common attack in the current Internet. Although ICN-based networks are expected to be less susceptible to this kind of attack (since no content flow occurs unless there exist explicit signaling denoting the demand as well as the availability of a specific content item) precautions should be considered. Tarkoma [12] investigated the threat of spamming in the context of publish/subscribe ICN architectures and identified three possible threats: bogus RENEs, publication replication, and subscribers interest prediction. In order to fight spamming, Tarkoma proposed the usage of a black list-based solution: every owner is identified by a private/public key pair, which is used for signing content; every time a content item is received it is checked whether the public key of the owner is contained in a black list of spammers.

A usual type of attack for spreading malicious content in the current Internet is through cache poisoning. Since the role of caching is expected to be leveraged in ICNs, this type of attack is expected to have bigger impact. Xie et al. [63] investigated this attack in the context of the NDN ICN architecture and proposed a mechanism for excluding malicious content from being cached, titled *CacheShield*. Their solution is based on the fact that legitimate content subscriptions follow a Zipf distribution, whereas subscriptions for malicious content (that are usually initiated by the attackers or by bots that are under the attacker's control) follow a uniform distribution. With this in mind, they proposed a mechanism that is implemented in the caches and decides, based on a record of requests, whether a piece of content should be cached or not.

Chapter 4

New security solutions for ICN architectures

4.1 Access Control

In ICN architectures efficient content dissemination is expected to be supported by dispersing content items in many network locations using, for example, caches, CDN-like networks, and bittorrent-like systems. Nevertheless this content dispersion raises severe security concerns, as it makes difficult to enforce access control policies. The solutions discussed in Section 3.1.1 assume a "closed-network" where all publishers are under the administration domain of the content owner, which however is not the general case. Therefore, a new solution is required that will facilitate access control on items stored outside the administrative realm of the content owner.

It is unrealistic to expect that each content item will be accompanied by an access control policy that each publisher should implement: not only that requires the existence of a complex access control mechanism at each publisher, but it also implies that everybody should have access to the user management system of the content owner. Access control policy computations should not be a task performed by any publisher that potentially hosts a content item, instead publishers could delegate access control decisions to third parties trusted by the content owners;

publishers then only have to respect the decision of these third parties and enforce them. To this end, we develop an access control enforcement delegation system for ICN architectures. This system operates simply by exploiting the functions of the underlay architecture and provides subscriber credential protection, privacy preservation, and facilitates interoperability. Furthermore, in this system, publishers can evaluate a request for an item against an access control policy, without having access to the policy itself. The basic principle of this system is that a content owner attaches to every content item a pointer to the access control policy that protects that item, rather than the policy itself. Any publisher can challenge a potential subscriber to invoke the access control policy and based on the output, the publisher can decide whether or not the subscriber is eligible to access the protected item.

4.1.1 Design

The proposed system introduces an additional entity: the *Access Control Provider* (ACP). ACPs are used to store the access control policy that protects a content item. An ACP provides the means to owners for creating access control policies and is responsible for evaluating subscribers against these policies. An ACP can be for example a Lightweight Directory Access Protocol (LDAP) server or a social network. Access control policies define the attributes that a subscriber should have in order to access a content item. They can be regarded as a function that accepts as input a subscriber ID and a set of attributes, and outputs *True* if the subscriber satisfies the input attributes or *False* on the contrary. In the general case it is assumed that publishers[1] and ACPs belong to different administrative domains, as well as, that every subscriber can authenticate itself to the ACP. Both publishers and ACPs are considered to be honest but curious, i.e., they operate as expected, but try to obtain as much information regarding subscribers as possible. No trust relationship is needed between a publisher and an ACP, they should only agree on a (trivial) communication protocol. On the other hand

[1]Recall that publishers are network entities and they could be for example web server, caches, CDN servers, etc.

an owner should trust an ACP to properly apply an access control policy and publishers to operate according to the ACP decision. A content item can be encrypted. Encryption can act as a counter-incentive for a publisher to misbehave, as unauthorized subscribers will not be able to read an encrypted item, therefore the publisher will just waste resources for sending it. Finally, it is assumed that the underlay architecture assures data integrity, confidentiality and provenance, therefore transmitted messages can not be read or modified.

The system's main goal is to protect the subscriber's credentials and preserve its privacy. The publisher learns nothing about the access control policy that protects a content item and gains no information associated with the subscriber. Similarly, details about the content item for which a subscriber requests access remain hidden from the ACP. Moreover the system eliminates, as much as possible, the subscriber's intervention in the communication between the ACP and the publisher: subscriber intervention introduces significant security risks that make the implementation of such systems complicated.

Figure 4.1 illustrates the system entities and gives an example of their interactions. In the illustrated example a user (Owner A) is the owner of an audio file which she wants to share with her friends, as specified by an online social network (OSN), which in this case acts as the ACP. For efficiency reasons the user wants to use a content distribution network (CDN), which will become a publisher. Initially, Owner A creates a new access control policy which outputs *True* for all her friends in the OSN and stores this policy in the OSN (arrow 01). The OSN creates a unique URI (A-C URI) for this access control policy (which becomes known to the owner). As a next step, Owner A sends the audio file to the CDN, indicating at the same time that this file is protected by the A-C URI access control policy (arrow 02). A friend of Owner A, using a device named Subscriber A, requests access to this file (arrow 03). The CDN responds with the A-C URI and a session secret (arrow 04a). At the same time the CDN notifies the OSN that it expects somebody that holds the secret to invoke the A-C URI access control policy (arrow 04b). The user of Subscriber A is a client of the OSN, she authenticates herself providing the same time the secret (arrow 05). The OSN notifies the CDN that a

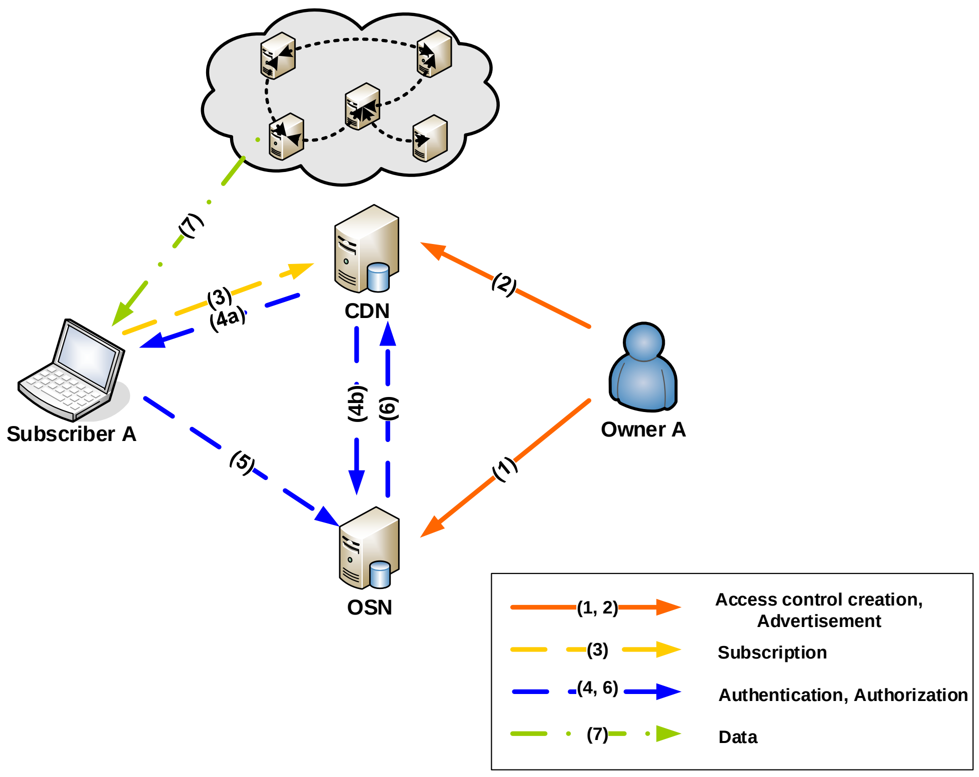


Figure 4.1: Access control system. CDN acts as a publisher and OSN as an ACP.

subscriber that knows the secret, authenticated correctly (06) and the CDN sends the audio file to Subscriber A (arrow 07).

It can be seen that the publisher (the CDN in our example) is completely oblivious about the content of the access control policy and the identity of Subscriber A. It has only access to the access control URI and to the end-point address of Subscriber A. Similarly the ACP (the OSN in our example) learns no information about the file in which Subscriber A is interested in. The publisher knows nothing about how the ACP implements access control. Finally, the publisher and the ACP have direct communication without the intervention of Subscriber A.

It should be noted here that since there is not an 1-to-1 relationship between an access control policy and an item, an access control policy is reusable, i.e., the same access control policy can be used for controlling the dissemination of multiple content items, belonging to many owners and provided by different publishers. Moreover an access control policy is not specific to a publisher, so in our example if the A-C URI was embedded in the audio file, any publisher that would receive that file–no matter from whom–would be able to protect this file using the exact same access control policy.

4.1.2 Implementation

An implementation of this scheme has been developed by extending a prototype of the PURSUIT architecture (discussed in Section 2.3), code-named Blackadder [64][2]. In order for a transaction to be completed the following steps are made: Initially, an owner creates an access control policy, denoted as F_s to describe the attributes that a subscriber should have in order to access an item protected by F_s. Let SId_{ACP} be a scope owned by an ACP. The owner creates a new scope, named SId_{fs}, under SId_{ACP}. In this new scope ("SId_{ACP}/SId_{fs}") everybody can advertise items, but only subscribers that abide by F_s are allowed to send subscription messages. An owner can create numerous policies using the same process. As a next step, the owner incorporates a meta-data field in the content

[2]The latest open-source version of Blackadder can be found at https://github.com/fp7-pursuit/blackadder whereas our implementation can be found at http://mm.aueb.gr/research/icn-access.zip

items that she wants to be protected by F_s. This field denotes that *this item is protected by "SId_{ACP}/SId_{fs}"*, i.e., the newly created scope. Therefore, the owner incorporates a "pointer" to F_s rather than F_s itself. If any owner wants to protect an information item with the same access control policy (i.e., F_s), she has simply to attach to that item the same pointer (e.g., in the form of a "meta-data" field).

Suppose that a publisher P_A owns scope SID_{rp} and it shares the protected item created previously (identified by ID_i). A subscriber S_A interested in accessing ID_i should send a subscription message for SID_{rp}/ID_i, that ultimately will reach P_A. P_A knows that ID_i is protected by "SId_{ACP}/SId_{fs}". It is also known that, by design, a subscriber that is able to complete the subscription process for an item published under "SId_{ACP}/SId_{fs}" is also allowed to send subscription messages for any item protected by "SId_{ACP}/SId_{fs}". Based on this, P_A, upon receiving the subscription message, generates a pseudo-item, identified by PI, `advertises` it under the scope "SId_{ACP}/SId_{fs}" and `notifies` the S_A that, in order for its subscription to be completed it has firstly to successfully subscribe to "$SId_{ACP}/SId_{fs}/PI$". Upon receiving this notification, S_A subscribes to "$SId_{ACP}/SId_{fs}/PI$". The ACP evaluates S_A's credentials against F_s; if the credentials are in compliance with F_s, the ACP sends a notification message to P_A informing her that a subscriber successfully subscribed to PI. However, since PI is only known to S_A, P_A will understand that the subscriber can access ID_i.

4.1.3 Evaluation

Security evaluation

The proposed system has the following security-related properties:

1. *Subscriber credentials are protected:* The only entity that has access to the subscriber credentials is the credentials provider, i.e., the ACP. Moreover due to the security primitives of the underlay architecture, an attacker cannot eavesdrop to the subscriber credentials, neither can she pretend to be an ACP.

2. *Subscriber privacy is preserved:* The only information a publisher learns

about a subscriber is an end-point address as well as that it is a client of an ACP. Similarly what an ACP learns about a subscriber is that it is interested in an item which a particular publisher hosts. An attacker eavesdropping on all communication channels can learn that an "endpoint" interacts with a particular publisher and a particular ACP. Only an ACP colluding with a publisher can obtain full information about a subscriber.

We also examine the following attack scenarios:
Man in the middle:

In this attack an attacker pretends to be a publisher, trying to hijack a session between a subscriber and a legitimate publisher, in order to obtain the information item that the subscriber requests. In order for this attack to be successful the attacker should persuade the subscriber that she is the legitimate publisher. However, the security primitives assumed for the underlay architecture assure data provenance, therefore the subscriber will understand that the messages received from the attacker did not originate from the legitimate publisher.
Malicious subscriber colluding with fake ACP:

In this attack a fake ACP tries to persuade a publisher that a subscriber successfully subscribed to the pseudo-item by sending a fake notification message. This attack cannot succeed due to the provenance assurance: the publisher will understand that the notification did not originate from the real ACP. An attacker may circumvent this security mechanism by "replaying" a captured legitimate notification. Providing that the publisher uses a different identifier for each pseudo-item it creates, the replay attack will also be unsuccessful, as the replayed notification will concern another pseudo-item ID.
Malicious subscriber colluding with a valid subscriber:

In this attack a malicious subscriber asks a valid subscriber to subscribe on her behalf to the pseudo-item. The RP will receive the notification and it will publish the protected item to the malicious subscriber. Therefore, this is a successful attack. Nevertheless this attack is equivalent to having a valid subscriber "giving" the protected item to an unauthorized subscriber using out of band mechanisms. We set aside this attack as an open issue, which currently should be handled by application-layer solutions (e.g., encrypt items using attribute-based encryption, therefore, the valid subscriber should have to reveal his private keys to the malicious user).

Communication overhead

The proposed system introduces a small amount of communication overhead to the content subscription process. For every content subscription, four new messages are introduced: the message for the pseudo-item advertisement, the notification sent from the publisher to the subscriber, the subscription to the pseudo-item and the notification sent from the ACP to the publisher. Various optimizations can be considered in order to decrease the communication overhead introduced in the subscription operation. E.g., the subscriber can decide what the identifier of the pseudo-item will be, therefore, the notification sent from the publisher to the subscriber can be omitted and the subscriber can send simultaneously the two subscription messages (one for the information item and one for the pseudo-item). In this case one message can be omitted.

4.2 Spam protection

It is generally argued that, by design, it is difficult to send spam traffic in systems based on ICN architectures, since no content flow occurs unless there exist explicit signaling denoting the demand, as well as, the availability of a specific content item. However, this observation is based on the assumption that subscribers express their interest explicitly in content identifiers.[3] In this section we consider the case in which subscribers request content items implicitly. In this setup a RN is responsible for finding the most appropriate content item that matches the subscription message. Therefore, the purpose of a spammer is to promote rogue items. To our knowledge this attack has not been previously discussed.

In order to prevent this attack we propose a solution based on our *inforanking* algorithm [65]. The proposed solution ranks content items based on the number

[3]E.g., in an email application, a user cannot subscribe to all email messages, not even to all senders.

of publishers that have advertised an item, as well as, based on subscriber feedback. Moreover, it is assumed that many publishers advertise the same item and the criteria that determine which items are the same are application specific. For example, in an application where subscribers send subscriptions for stock prices, two items are the same if they contain the same price, whereas in an application where subscribers send subscriptions for images from a specific event, two items are considered to be the same if the outputs of a *perceptual* hash function are very close. The proposed solution relies on the fact that malicious publishers will try to generate as many similar items as possible in order to circumvent *blacklisting.* By ranking items and not publishers, we discourage malicious publishers from taking advantage of legitimate publishers, e.g., by using viruses and worms.

4.2.1 Inforanking

Inforanking is a vote-based approach for ranking information items. It was initially developed for isolating polluted pieces of information in file sharing networks and its development was based on the observation that in these networks malicious users provide numerous polluted versions in order to avoid blacklisting. Its design was driven by the requirement to add the least possible overhead to the already deployed architecture. Inforanking has been proven to be more effective than user-ranking based solutions in terms of polluted objects isolation. Moreover, it has minimum impact on the system.

In inforanking, users may vote only positively regarding a specific information item. Moreover a user may vote only once. Each vote of a user U in a context R is weighted by a factor w computed as $w = 1/(\sum V_u)^a$ where $\sum V_u$ is the sum of U's votes in R and a is a fixed value. As an example consider the set of files of Table 4.2.1 which is composed of four items. This table shows how the score of each item is calculated (using $a = 1$). The first column of the table contains the identifiers of the items that are included in the set. The second column contains a list of users that have voted for each item and the third column contains the score of each item. As it can be seen, user U1 has voted for 4 items in the set, therefore his votes are weighted by 0.25. On the other hand users U5, U6, and U7 have

ItemID	**Users**	**Score**
Item01	U1, U2, U3, U4	$0.25 + 0.5 + 0.5 + 0.5 = 1.75$
Item02	U1, U2, U3, U4	$0.25 + 0.5 + 0.5 + 0.5 = 1.75$
Item03	U1, U5, U6, U7	$0.25 + 1 + 1 + 1 = 3.25$
Item04	U1	0.25

Table 4.1: Inforanking voting example. The more times a user votes, the less significant his vote is.

voted for a single item, so their votes are weighted by 1.

4.2.2 Design

The goal of this system is to enable RNs to isolate spam items. Inforanking is applied to the set of items that match a subscription. With inforanking, spam items receive **lower** score. The score of an item i is calculated using a two step approach. In the first step, the number of publishers that have advertised the item is considered (this is the publisher based score of i, PR_i) and in the second step the subscribers' votes are considered (this is the subscriber based score of i, SR_i). The rule of thumb for the first step is that items that are advertised by many well-behaved publishers are probably valid. Well-behaved publishers are those who advertise a "normal" number of items. Inforanking assures that the bigger the number of items a publisher advertises the smaller is the effect it has on the item's score. The fact that a publisher advertises an item is considered as a positive vote. This vote is weighted by a factor that depends on the total number of items the publisher has advertised and are included in the set of items that match the subscription message. The publisher-based rank of an item i in a set R is $PR_i = \sum PV_R$, where PV_i is a vote for i by a publisher in R. An advertisement of i by a publisher P is cosidered as a vote and it is weighted by $1/\sum Adv_P$ where Adv_P is an advertisement of P in R. As it can be observed, inforanking is used with $a = 1$. As an example, if an item i is advertised by two publishers and each of these publishers has advertised 4 other similar items, then the publisher-based rank of i is $PR_i = (1/4 + 1/4) = 0.5$. Publisher-based ranks are then normalized using the following formula: $NPR_i = PR_i/\sum P$ where $\sum P$ is the total number of publishers i=that have advertised items in R. During this first step, the score of an information item is calculated based on data and functionality provided by the ICN architecture, i.e., no extra state or communication overhead is added to the network.

During the second step subscribers vote for non-spam items, i.e, whenever a subscriber receives a non-spam item, it informs the RN. Subscribers may vote only once for a specific item and there is no vote that indicates that an item is spam. Every vote SV of a subscriber S in a set R of items that match a subscription, is weighted by $1/\sum SV_S$ where $\sum SV_S$ is the number of votes of S in R. So, if an item i has received two votes from two different subscribers and each of these subscribers has already voted for 10 similar then the subscriber based rank of i is $SR_i = (1/10 + 1/10)$. The subscriber-based ranking is also normalized using this formula: $NSR_i = SR_i/\sum S$ where $\sum S$ is the total number of subscribers that have voted in R. This step requires some additional state and communication overhead: each RN should maintain a list of votes and each vote is an extra message in the network. Nevertheless the state is fully distributed to all RNs and the vote message may possibly be encapsulated in other messages, e.g., in an ACK message.

The final score of an item i is $IR_i = NPR_i + NSR_i$ and the item chosen by the RN is the one with the highest score.

4.2.3 Evaluation

The proposed solution is evaluated using two threat models. The first threat model concerns an architecture where malicious publishers advertise spam items. In the second threat model, in addition to the malicious publishers, malicious subscribers who collude and vote against valid items are considered.

Simulation Setup

Using OMNeT++ [4], a network consisting of 50 legitimiate publishers and 200 legitimate subscribers is simulated. Legitimate publishers advertise 1 information item. This item is selected from a pool of 5 valid information items, using a

[4]http://www.omnetpp.org

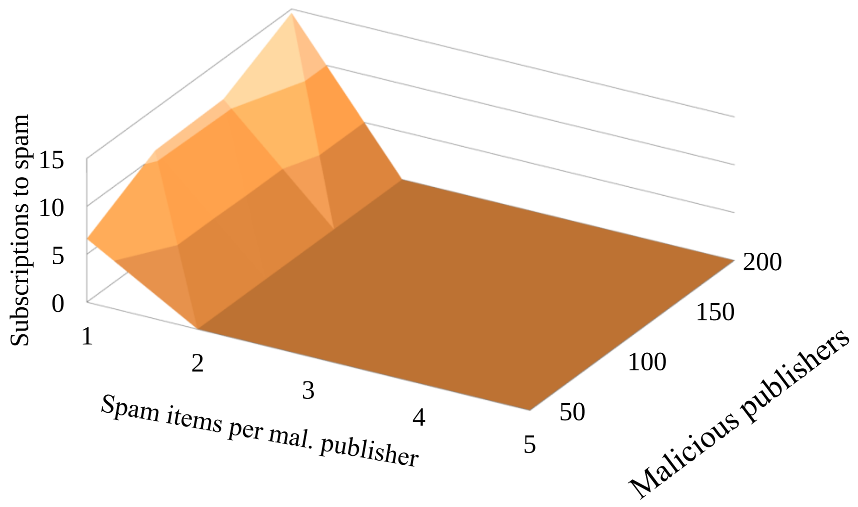


Figure 4.2: Total subscriptions that lead to spam items as a function of the number of malicious publishers and the number of items each malicious publisher advertises.

Zipf distribution. All the advertised information items are similar and all advertisements are stored in the same RN. Subscribers send subscription messages at specified time intervals. The simulation ends when all subscribers obtain one valid content item.

Threat model A

We now examine the case in which 50, 100, 150, and 200 malicious publishers exist in the netwrok. Malicious publishers advertise a number of items from a pool of 100 spam items. Figure 4.2 shows the total number of subscriptions that lead to spam items as a function of the number of the malicious publishers and the items they advertise.

As it can be seen the number of subscriptions that lead to spam is zero when malicious publishers choose to advertise more than one spam item. Moreover, it can be observed that this number does not increase proportionally to the number of malicious publishers.

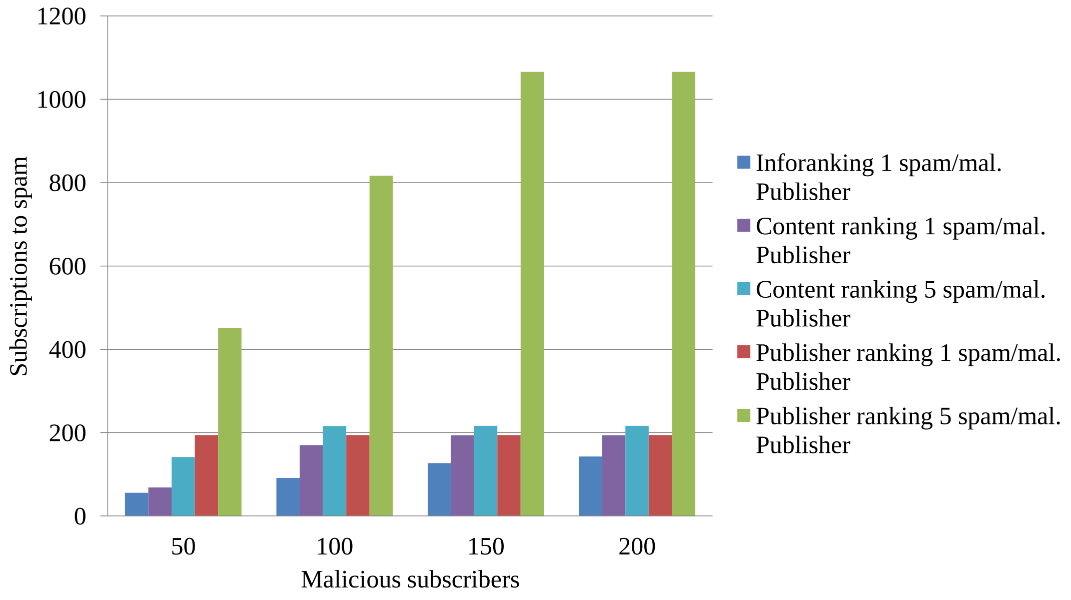


Figure 4.3: Total subscriptions that lead to spam in a network in which 50% of the publishers are malicious and malicious subscribers collude.

Threat model B

In this threat model malicious subscribers are also considered. We compare our solution against two other approaches: (i) a publisher-based ranking approach and (ii) a content-based ranking approach that considers both positive and negative votes. In the publisher-based ranking approach subscribers vote in favor of or against publishers. In this case malicious subscribers try to promote malicious publishers. Similarly in the content-based ranking approach subscribers vote in favor of or against content items and malicious subscribers try to promote spam items. In all experiments 200 malicious publishers are considered. Each of the malicious publishers advertises 1 or 5 spam items chosen from a pool of 100 spam items. It should be noted here that when inforanking is used, if malicious publishers advertise 5 spam items, no subscription will lead to spam, therefore this case is not considered.

Figure 4.3 shows the total number of subscriptions that lead to spam for each approach. As it can be seen inforanking is more effective compared to the other solutions.

4.3 User privacy

One of the biggest privacy challenges in a networking architecture is to provide end-user *unobservability.* In the context of ICN this translates into enabling users to subscribe for content items and receive them without anybody–not event the publisher–learning which items the subscriber requested. The solution of Arianfar et al. [53] presented in Section 3.2.1 contributes to this direction. However, this solution adds significant network overhead since periodically, many advertisements have to be sent. Moreover, the owner has to perform many computations and permanently store the results, in order to achieve a significant level of privacy.

A trivial solution to solve this problem is to use a brute force approach and return to the subscriber all the items that a publisher stores. We argue that two main reasons could stand against this simplistic approach. First, it is often not an option due to the size of the items. Secondly, the business model of the publisher (or the owner) might rely on a per-item-retrieval charge, therefore, such a solution would negate this business model. Another approach for achieving this goal is to use *Private Information Retrieval* (PIR). PIR is a query-response paradigm that allows the retrieval of a record from a database without revealing any information about the retrieved record, not even to the database (see [66] for a survey on this topic). Although promising, PIR, relies on cryptographic primitives that impose significant computational overhead and, therefore, can be practically applied only for small data (in the order of some bytes).W propose an optimized PIR protocol that protects the resolution part of the subscription operation. In the proposed solution the following properties hold: (i) a subscriber is able to subscribe for a content item without needing to reveal his choice to the RENE, (ii) the RENE responds back to the subscriber with the RN that has a matching entry for the item of interest without being able to determine the exact item that matched. Consider for example the simplified RENE of Figure 4.4. All RNs maintain a list of *pointers* to the (network) location of their children (not all lists are shown in this figure). Suppose that a subscriber wants to subscribe for an item with identifier prefix $A/D/E/B$. With the proposed solution the subscriber will learn in a privacy preserving way the location of the RN in the red circle, which is responsible for

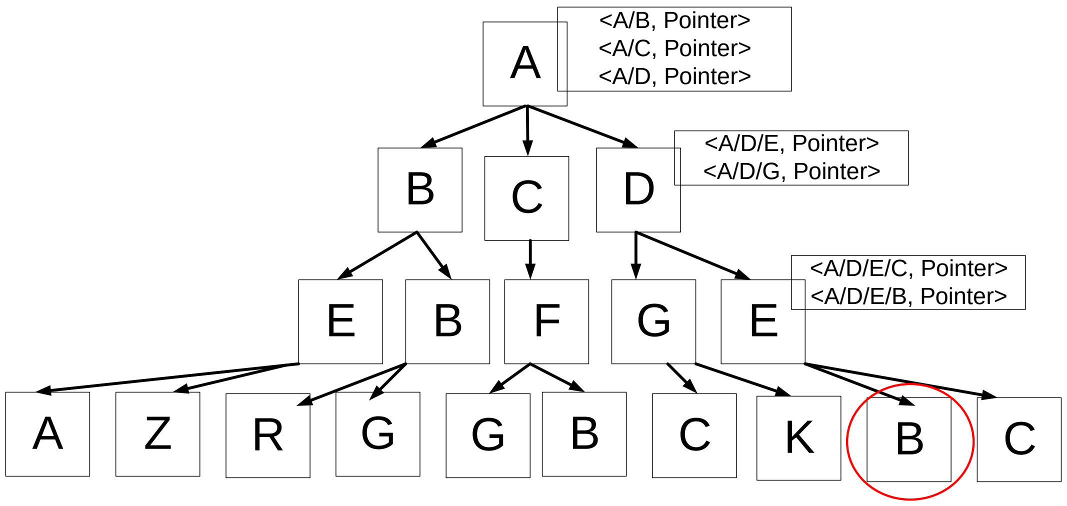


Figure 4.4: A RENE example. Each box represents a RN. RNs are hierarchically organized and the path to a RN denotes the identifier prefix that this RN manages. Each RN maintains a list of “pointers” to its children. The circle shows the RN that has a record that matches a subscription.

managing the desired item. The solution is applied to all RNs that are located up to one level higher than the desired RN and it is based on the Paillier cryptosystem.

4.3.1 The Paillier cryptosystem

The Paillier cryptosystem [67] is a public-key based approach. The public key K_{pub} is a pair of numbers[5] (n, g), where n is the product of two large primes (p, q) and g is a random number in $\mathbb{Z}^*_{n^2}$. The private key K_{prv} is the least common multiple of $(p-1)$, $(q-1)$. The encryption function $E(m)$ of a message $m \in \mathbb{Z}_n$, uses as input the public key K_{pub} and a randomly selected number $r \in \mathbb{Z}^*_n$. The resulting ciphertext c, belongs to $\mathbb{Z}^*_{n^2}$, i.e., the ciphertext is twice as large as the plaintext. For the decryption $D(c)$ of the ciphertext, only the private key, K_{prv}, is required, i.e., the random number r is not used during the decryption.

The most interesting property of the Paillier cryptosystem is its homomor-

[5]For the homomorphic operations of our scheme only n is required [68].

phism, i.e., let $a, b \in \mathbb{Z}_n$, then:[6]

$$E(a) \cdot E(b) = E(a + b)$$

Moreover, it is possible to multiply an encrypted number a with a known number b, without revealing a or the result; given $E(a)$ and b, $E(a \cdot b)$ can be calculated as follows:

$$E(a \cdot b) = E(a)^b$$

4.3.2 Design

In this section details about our scheme are provided. It is assumed that a subscriber can learn, using an out-of-band mechanism, an ordered list of all the identifier prefixes that a RENE manages (or in case this information is confidential or the size of the list is big, a list of their hashes). Moreover, it is assumed that all pointers are of equal size. Finally it is considered that all identifiers belong to $\mathbb{Z}_n$.

Initially a simple model is considered. In this model the RENE is treated as a single entity, which manages all the prefixes. Then, by taking advantage of the hierarchical organization of the RENE, improvements are proposed.

A PIR model

We define a PIR model is defined, based on [13]. This model is composed of the following functions:

CreateQuery(ID, S)
The *CreateQuery* function is executed by a subscriber in order to construct a subscription message. This function takes as input an ordered list $ID = id_1, id_2, ..., id_m$ of the prefixes managed by the RENE and a corresponding set of numbers $S = s_1, s_2, ..., s_m$. From all items in ID, the subscriber is interested in only one of them. The values of S are chosen in such a way that the result of the linear expression $s_1 \cdot id_1 + .. + s_m \cdot id_m$ is the prefix of the identifier of the item in which the subscriber is interested. The set S is trivially constructed: if id_l is the identifier of the item in

[6]All operations are $mod\ n^2$.

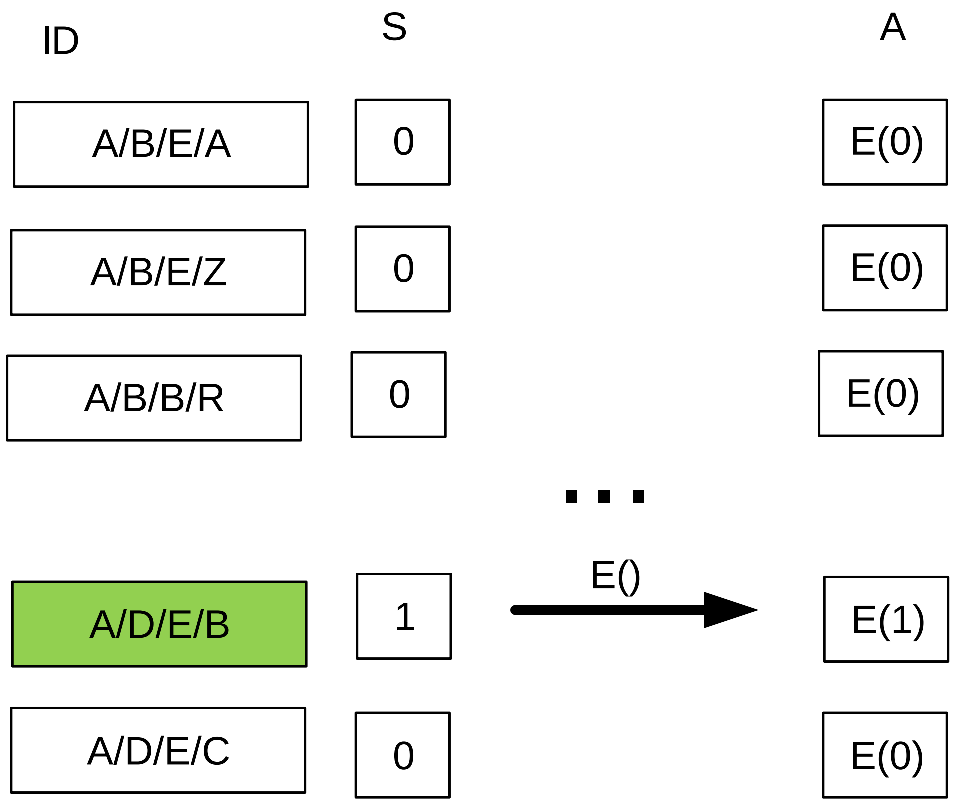


Figure 4.5: Lookup message creation. The subscriber is interested in the second from the bottom prefix of the ID set, therefore, the corresponding element of the set A is E(1).

which the subscriber is interested then $s_l = 1$ and $s_i = 0, i \neq l$. The function generates a public/private key pair, denoted as K_{pub}/K_{prv}, and outputs a subscription message $Q = \{K_{pub}, A\}$,[7] where $A = E(S)$ is a new set which is constructed by encrypting all elements of S using K_{pub}, i.e., $a_1 = E(s_1)$, $a_2 = E(s_2)$,.., $a_m = E(s_m)$. Note that due to the probabilistic property of the Paillier cryptosystem if $s_x = s_m$ then $a_x \neq a_m$. Figure 4.5 illustrates how the set A is constructed.

CreateResponse(Q, P)

The CreateResponse function is executed by the RN when a subscription message

[7] K_{pub} includes only the n part of the key.

$Q = \{K_{pub}, A\}$ is received. It takes as input Q and the set ID used for the Q construction. The function outputs a response R by exponentiating each element a_i of the set A to the pointer that corresponds to the i^{th} identifier of the ID set – denoted as $pointer_i$ – and by multiplying all the exponents. Assuming that the size of a pointer is less than n, R is calculated as follows:

Algorithm 1 CreateResponse(Q, ID)

```
$R = 1$
for each $id$ in $ID$ do
    /* $a_i$ is the $i^{th}$ element of set A included in Q */
    $R = R * (a_i^{pointer_i}) \bmod n^2$
end for
```

If the size of a pointer is bigger than n then, each pointer is divided in blocks of size n and the above algorithm is accordingly adapted. Due to the Paillier cryptosystem's homomorphism properties, this algorithm outputs the result of the following expression:

$$(a_1)^{pointer_{id_1}} \cdot ... \cdot (a_m)^{pointer_{id_m}} =$$

$$E(pointer_{id_1} \cdot s_1) + ... + E(pointer_{id_m} \cdot s_m)$$

Although R is generated by combining all pointers, its size is the size of a single encrypted pointer. Figure 4.6 illustrates how R is constructed based on our example.

ReadResponse(K_{prv}, R)
The ReadResponse function is executed by a subscriber upon receiving a response R. This function decrypts R using the private key K_{prv}, generated by the CreateQuery function, and returns the pointer to the RN that has a matching entry for the item of interest.

PIR subscriptions over a pseudo flat RENE

Now the PIR model is expanded in order to be applicable to a hierarchical RENE.

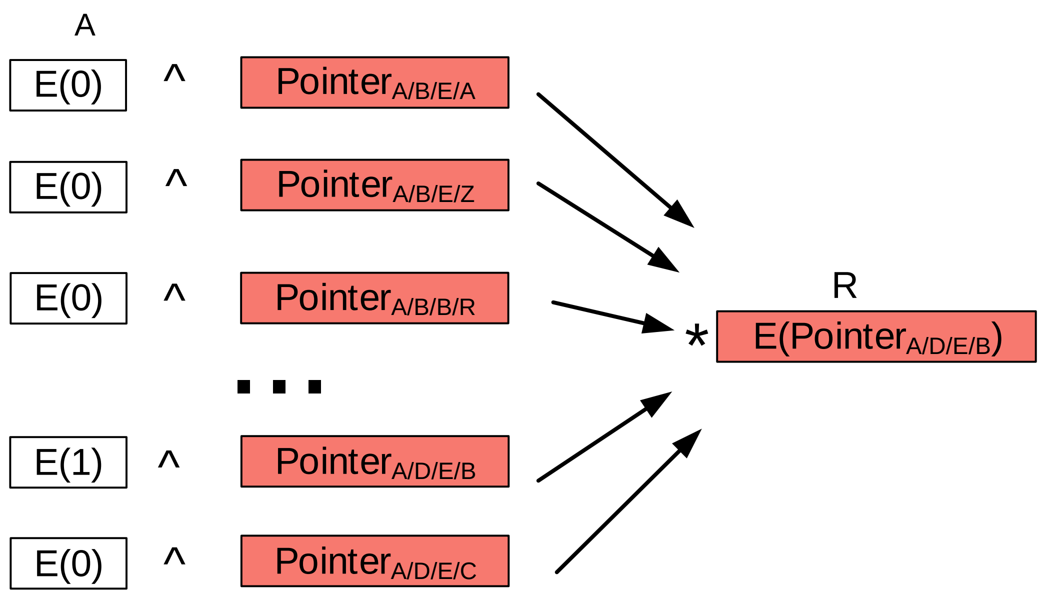


Figure 4.6: Response creation. The symbol $\wedge$ denotes exponentiation and the symbol $*$ denotes multiplication.

The subscriber still treats the RENE as being flat and constructs a subscription message over the (flat) information space. The query is sent to the root RN of the RENE. The root RN splits the message and forwards to each of its descendant RNs the corresponding part. This procedure is recursively repeated until all RNs up to one level higher than the desired RN receive the part of the query that corresponds to them. Then each RN at that level calculates a response and forwards it to its parent. Each RN multiplies the responses it receives from its children and forwards the product to its parent. This procedure is repeated until the root RN receives the responses from all of its children. Finally, the root RN forwards the response to the subscriber. Figure 4.7 illustrates this approach.

Lookups over hierarchically organized RENE

A first approach for implementing subscriptions over a hierarchically organized RENE is based on the fact that given an ordered list of prefixes managed

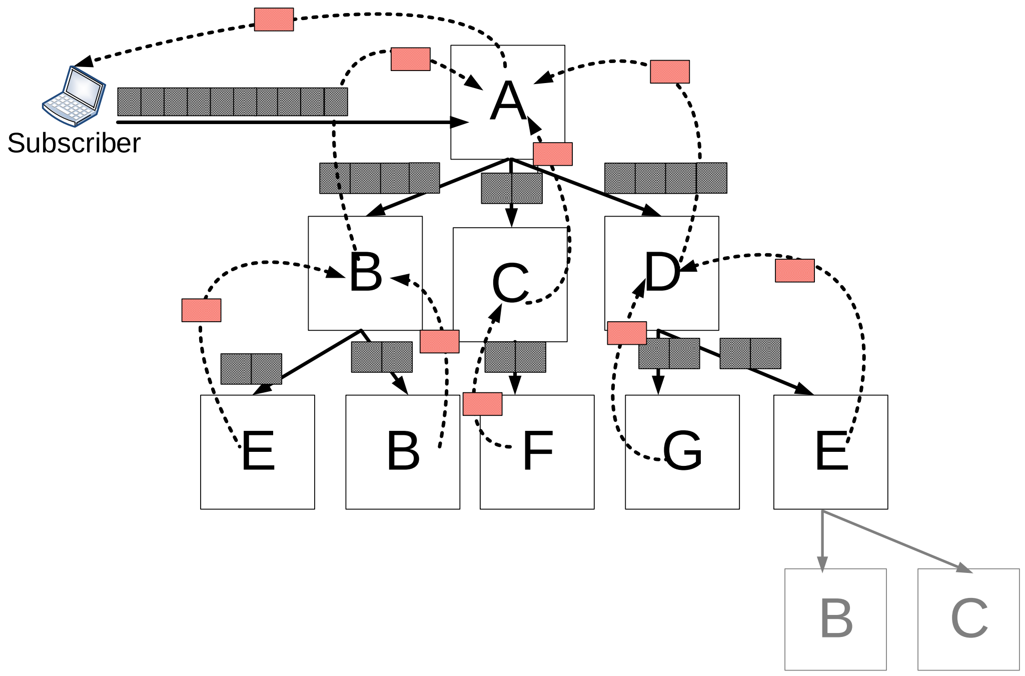


Figure 4.7: PIR subscription over a pseudo flat RENE. The query is as big as the number of the records of the leaf nodes. Each leaf node calculates its portion of the response. Responses are multiplied and forwarded to the root RN.

by the RENE, it is possible to reconstruct the RN hierarchy. The subscription message is now constructed using the following procedure: the subscriber considers that each level of the RENE hierarchy contains only the RN that manages the part of the identifier prefix of interest, then for each level of the hierarchy (up to one level higher than the desired RN), constructs a sub-message that filters the pointers maintained by the assumed unique RN; if not all RNs of the same level have the same number of pointers, then the subscriber considers that the assumed unique RN has as many pointers as the RN that has the maximum number of pointers at this level. When done all sub-messages are concatenated. Consider for example the RENE of Figure 4.8. Suppose that a subscriber is interested in the content item with prefix $A/D/E/B$. Initially, the subscriber constructs a sub-message that filters all but the third pointer of the RN that manages B; this sub-message contains a set A of the form E(0), E(0), E(1). Similarly the subscriber, treats the second level of the RN as being composed only by the RN that manages D. Moreover, the subscriber considers that this RN has two pointers to other RNs and creates a message that filters all but the second pointer, i.e., the message contains a set A of the form E(0), E(1). Finally, the subscriber treats the third level of the RENE as being composed only by the RN that manages E, and creates a sub-message that filters all but the second pointer of this RN, i.e., this sub-message contains a set A of the form E(0), E(1). The subscriber concatenates the sub-messages and sends the result (i.e., a subscription message with an A set of the form E(0), E(0), E(1), E(0), E(1), E(0), E(1)) to the root RN. The root RN keeps the part of the subscription message that corresponds to it and forwards the rest to its children. These RNs in return keep the part of the message that corresponds to them, and forward the rest to their children. This procedure is repeated until all the RNs up to one level higher than the desired RN receive their part of the message. In our example, this procedure will result in the RN that manages A having received a subscription message with an A set of the form E(0), E(0), E(1), the RNs that manage A/B, A/C, and A/D, a message with an A set of the form E(0), E(1) and the RNs that manage $A/B/E$,...,$A/D/E$, a message with an A set of the form E(0), E(1). Each RN applies the part of the subscription they hold to the pointers

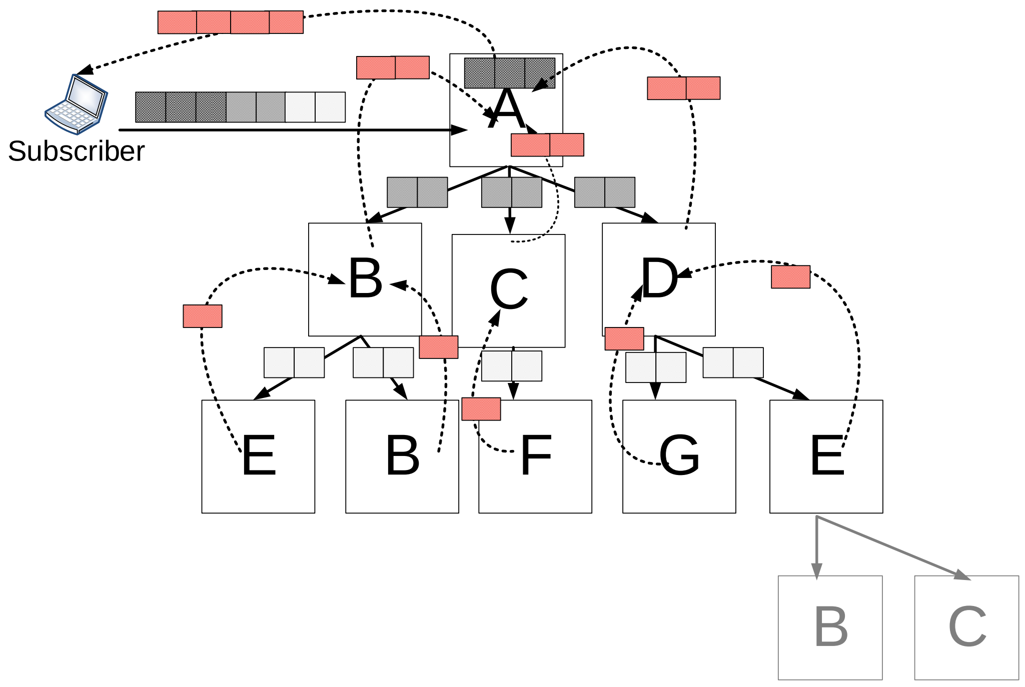


Figure 4.8: PIR subscription over a hierarchical RENE. Queries are smaller but responses are larger.

they maintain and forward the result to their parents. The parent RNs, apply the part of the subscription they hold to the results they receive from their children and forward the outcome to their parents, and so forth. The output of the root RN is the desired pointer encrypted as many times as the levels of the hierarchy. It should be noted here that each intermediate RN, before applying the part of the subscription it holds to the results it receives from its children, it has to split the result in blocks of maximum size n. Then it should use the same element of A for all blocks of the same result. This is a procedure similar to the one followed in [69].

A second approach that can be used in order to implement subscriptions over a hierarchically organized RENE is a variant of the PIR approach. In this case the RNs, up to one level higher than the desired RN, instead of sending their responses to their parent RN, they send them directly to the subscriber. In this case since the subscription is only applied to these RNs, its A set has to be as big

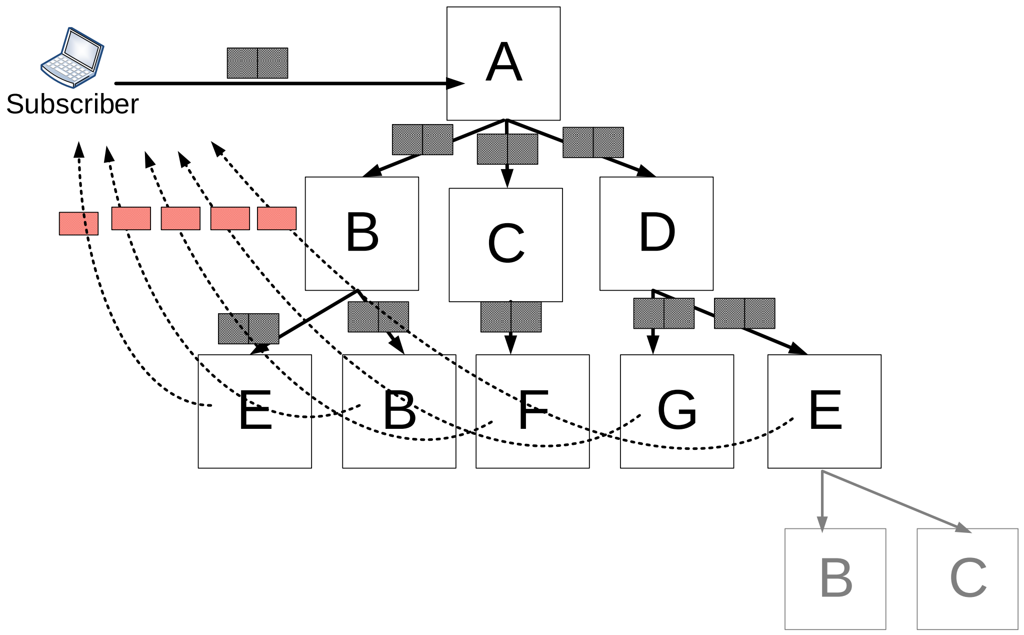


Figure 4.9: PIR subscription over a hierarchical RENE where leaf RNs reply directly to the subscriber.

as the number of the records, of the RN at this level with the maximum number of records. This procedure is depicted in Figure 4.9. With this approach a subscriber will receive some redundant responses, which can be disposed.

4.3.3 Evaluation

The proposed system is implemented using the Blackadder prototype. The Paillier cryptosystem is implemented, using the advanced crypto software collection [70].

Security Analysis

In this section we analyze the security properties of the proposed system. Initially a threat model is defined, and then the system is evaluated based on this model.

Threat model

For the security analysis a threat model it is assumed in which an adversary wants to learn information about a subscriber's preferences. It is assumed that a security attack is successful when an adversary learns, without being detected, some information about (i) the item in which a subscriber is interested, or (ii) the items in which a subscriber *is not* interested. In this threat model the following type of adversaries are considered:

- Passive third party adversary (eavesdropper).
- Active third party adversary. This is a third party adversary that may modify transmitted packets.
- Honest-but-curious RN. This is a RN interested in learning subscriber preferences, but without deviating from the specified protocol.

Security evaluation

Initially the case of an eavesdropper that learns the response of the RENE is examined. All responses are Paillier ciphertexts and since the Paillier cryptosystem is semantically secure [67] an adversary can not deduce any information. All responses are encrypted with a public key generated by the subscriber, therefore given two responses for the same identifier, but targeting different subscribers, an adversary is not able to tell if they concern the same identifier or not. Moreover, since the Paillier cryptosystem is probabilistic, two responses that concern the same identifier, targeting the same subscriber, will differ from each other. An active third party may be able to tell if two different responses targeting the same subscriber concern the same content item simply by discarding the second response and by repeating the first one; if the subscriber proceeds as no error has occurred, then these two responses concern the same item. A solution that can be used by a subscriber in order to mitigate this attack is to record all the responses (or their hashes) that have been received, and discard all duplicates. An honest-but-curious RN has the same capabilities as an eavesdropper.

We now examine the case in which an eavesdropper intercepts a subscription message. The subscription messages of the PIR model, as well as the second type of subscription over a hierarchical information space, are adaptations of [13] in a hierarchical RENE, therefore, their security against eavesdroppers can be deduced from that work. Similarly, the first type of subscription over a hierarchical information space can be proved to be secured against eavesdroppers using the work in [69]. An active third party may be able to learn some information about the items in which a subscriber is not interested in by interchanging two elements of the set A of a subscription message; if the subscriber proceeds and no error has occurred, then it means that these two elements are encryptions of zero, therefore, the subscriber is not interested in the corresponding content items. In order to prevent this type of attack, the subscription message should be digitally signed by the subscriber (or it should be encrypted using an authentication encryption mechanism and a key shared between the subscriber and the RENE) and the RENE should be honest in informing the subscriber if the subscription has been tampered. An honest-but-curious RN has the same capabilities as an eavesdropper.

The case of malicious RNs

In this sub-section a case which is not considered in the threat model is discussed: the case of malicious RNs that deviate from the specified protocol. As it will be seen, when a malicious RN deviates from the specified protocol it is possible to learn some information about the subscriber's *non-preferences*. All the following attacks are based on the fact that it is hard to tell if a RN has used all the available data in order to create a response and a solution for these attacks has been left as future work.

The first attack concerns deviation from the response creation procedure. In the PIR subscription type, as well as in the first type of subscriptions over a hierarchical information space, it can be observed that if the intermediate response of a RN, that *does not* manage part of the identifier of the item in request, is discarded then (i) the final response will still be valid, and (ii) the subscriber will not be able to tell that the response has been manipulated. The same applies if a RN up to one level higher than the desired RN does not include in the response calculation a pointer that is not of interest to the subscriber. Therefore, by discarding an intermediate response or a pointer and by observing if the subscriber proceeds as if no error has occurred, then a malicious RN is able to tell some of the identifiers in which the subscriber is not interested. When the second type of subscription over a hierarchical information space is used, this attack can only by launched by the RNs up to one level higher than the desired RN by not considering some of their pointers during the response generation.

The next attack is realized when a malicious RN manipulates a subscription. If the PIR model or the first type of subscription over a hierarchical information space are used, a malicious RN can omit to forward to a subsequent RN the corresponding part of the subscription message, excluding this RN–and its children–from the response creation procedure. If the subscriber proceeds as if no error has occurred, then the malicious RN is able to tell some of the identifiers in which the subscriber is not interested. When the second type of subscriptions over a hierarchical information space is used and if the subscriber knows the expected number of responses, then this attack is not applicable. However, in this case a malicious RN may alter the subscription message it forwards to its children (instead of omitting it). Again, if the subscriber proceeds as if no error has occurred, then the malicious RN learns some of the identifiers in which the subscriber is not interested. Nevertheless, this attack can be mitigated using digital signatures.

Variations of these attacks are applicable to many PIR systems, including [69] and [13] and to our knowledge no solution has yet been provided.[8]

Analysis of the Introduced Overhead

Throughout this section the notation of Table 4.2 is used.

Communication overhead

Assuming a public key of $Size_{Pub}$ bits,[9] the overhead introduced in the communication between a subscriber and a RENE is now calculated. In the PIR model the size of a subscription message is $Size_{Pub} + |ID| \cdot 2 \cdot Size_{Pub}$ bits, where $2 \cdot Size_{Pub}$ is the size of the encryption of 0 or 1. The size of the response is

[8]Reference [71] provides a solution that assures that a *subject* includes in the response calculation a *property* that it really owns, however this solution does not tackle the case in which the *subject* does not include a (valid) *property* in the response calculation.

[9]Considering only the n part of the key.

Table 4.2: Notation

$\lvert ID \rvert$	The size of the identity space managed by the rendezvous system
$Size_p$	The size of a pointer
h	The height of the rendezvous system hierarchy
$max(p_i)$	The number of pointers of the RN with the maximum number of pointers at level i
$avg(p_i)$	The average number of pointers of a RN at level i
$\lvert leaf \rvert$	The number of RNs one level higher than the desired RN

$2 \cdot \lceil Size_p / Size_{Pub} \rceil \cdot Size_{Pub}$. In the first type of subscriptions over a hierarchical RENE, the size of a subscription message is $Size_{Pub} + \sum_{i=0}^{h-1} max(p_i) \cdot 2 \cdot Size_{Pub}$ bits. The size of the response is $2^h \cdot \lceil Size_p / Size_{Pub} \rceil \cdot Size_{Pub}$. In the second type of subscriptions over a hierarchical RENE the size of a subscription message is $Size_{Pub} + max(p_{h-1}) \cdot 2 \cdot Size_{Pub}$ bits, where $max(p_{h-1})$ is the number of pointers of the RN with the maximum number of pointers, located one level higher than the desired RN. In this type of subscriptions a subscriber receives as many responses as the RNs one level higher than the desired RN, with each response having size $2 \cdot \lceil Size_p / Size_{Pub} \rceil \cdot Size_{Pub}$. Therefore, the RENE to subscriber communication overhead is $|leaf| \cdot 2 \cdot \lceil Size_p / Size_{Pub} \rceil \cdot Size_{Pub}$.

The PIR model experiences the biggest communication overhead and this happens because it treats the RENE as being flat. PIR optimizations are also applicable in this system, but they have to be applied per RN. Therefore, if we consider the optimization proposed in [69], according to which the (per-RN) information space is organized in a 2-hypercube, then the subscription message size of this optimized PIR model would be $|leaf| * log_2(avg(p_{h-1}))$ and the response size would be $4 \cdot \lceil Size_p / Size_{Pub} \rceil \cdot Size_{Pub}$.

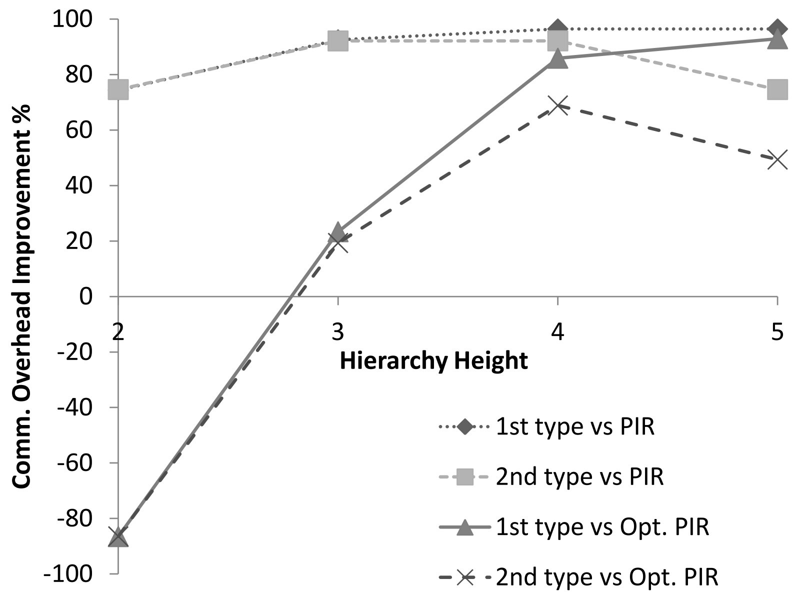


Figure 4.10: Communication overhead improvement, compared to PIR over a flat RENE, as a function of the RENE hierarchy height.

The improvement of the communication overhead due to the proposed approach is now evaluated. A RENE with varying height and branching factor is considered. Moreover the inter-RN communication overhead is considered to be negligible. The RNs maintain as many records required in order for $|ID|$ to be 1000. Moreover it is assumed that a single pointer fits to an encryption block and the size of the public key is 1024bits. Figure 4.10 shows the communication overhead improvement when the branching factor is 4 and h varies from 2 to 5. Figure 4.11 shows the communication overhead improvement when h is 4 and the branching factor varies from 3 to 6. In both cases the communication overhead between a single subscriber and the RENE is examined. As it can be seen the only case in which the proposed system experiences larger communication overhead is when the RENE is almost flat (i.e., h is 2). Moreover it can be seen that when the RENE is highly distributed, the communication overhead improvement is almost 97%.

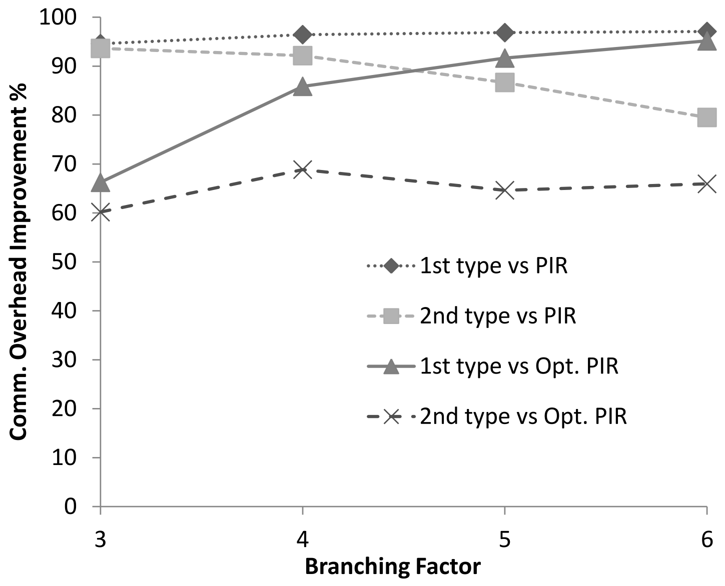


Figure 4.11: Communication overhead improvement, compared to PIR over a flat RENE, as a function of the RENE hierarchy branching factor.

Computational overhead

The use of cryptography introduces computation overhead to both the subscribers and the RENE as these entities have to perform multiplications and exponentiations using large numbers. In an Ubuntu 12.04 based machine, using an Intel i5 processor at 2.8 GHz with 4GB of RAM and the GMP library ver. 5.1.0, the modular multiplication of two 256-bytes numbers was performed in $2\mu s$ and the modular exponentiation of a 256-bytes number raised to a 128-bytes number was performed in $7.5ms$. The encryption of a number (i.e., the creation of an element of the set A of a subscription message) was performed in $9.6ms$. Nevertheless, this is a computation that can be made offline, i.e., a subscriber can pre-calculate an arbitrary number of encryptions of 0 and 1.[10] The decryption of a 256-bytes ciphertext was performed in $7.5ms$.

Assuming that a pointer is less than n, in the PIR model all RNs located one level higher than the desired RN, have to perform as many exponentiations as the number of pointers they maintain and all RNs have to perform as many multiplications as the number of pointers they maintain. In the first type of subscriptions over a hierarchical RENE all RNs located up to one level higher than the desired RN have to perform as many exponentiations and multiplications as the number of pointers they maintain. A RN at level i will receive $avg(p_i)$ responses of size $2^{h-i-1}Size_{Pub}$. Each of these responses has to be split in 2^{h-i-1} blocks of size $Size_{Pub}$ and for each block an exponentiation and a multiplication has to be performed. A subscriber in order to decrypt the final response has to perform $\sum\limits_{i=0}^{h-1} 2^i = 2^h - 1$ decryptions. Finally, in the second type of subscriptions over a hierarchical RENE all RNs located up to one level higher than the desired RN perform as many exponentiations and multiplications as their records.

It should be noted that the time required to create a response is not proportional to the number of the operations, since all operations of the same level can be done in parallel. Therefore, if the time of a multiplication is considered to be negligible, the total time required to create a response is the same whether the

[10]Note however that the same encryption should not be used in two different messages in order to avoid pattern based attacks.

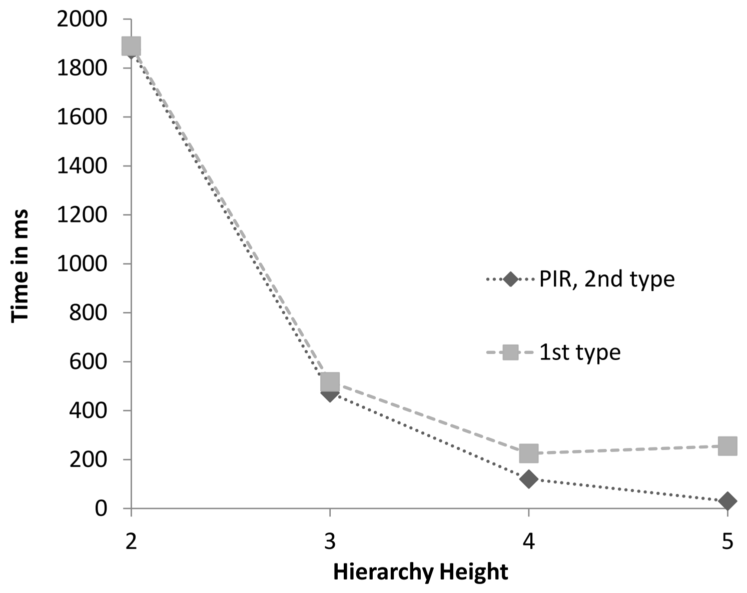


Figure 4.12: Response calculation time with varying RENE height.

PIR model or the second type of subscriptions over a hierarchical RENE is used. Figures 4.12 and 4.13 show the estimated response calculation time based on the number of exponentiations and the aforementioned property, using the setup of the previous section. In Figure 4.12 the branching factor is 4, and in Figure 4.13 the height of the RENE is 4. As it can be observed the more distributed the RENE is, the less time is required for the response calculation. The reason for that is that the more distributed the RENE is, the more RNs work on the response, with each of them performing calculations on less data.

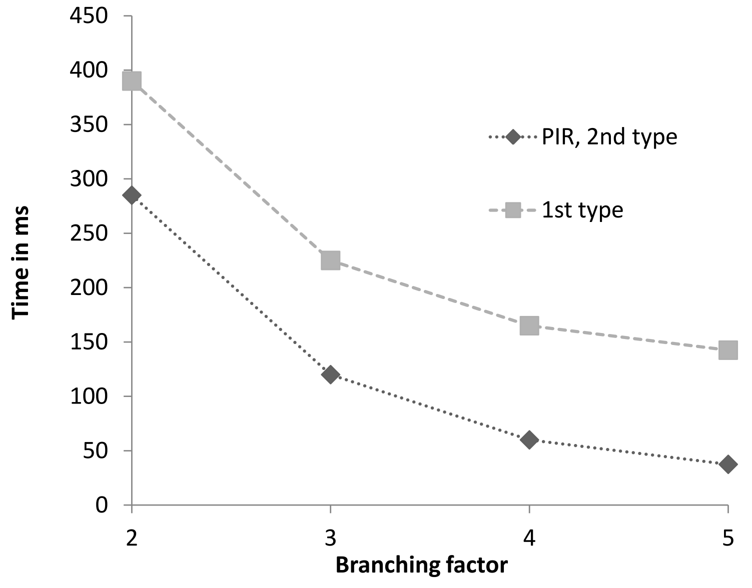


Figure 4.13: Response calculation time with varying RENE branching factor.

Chapter 5

Applications of ICN specific security solutions in the current Internet

5.1 Fighting content pollution in P2P file-sharing networks

P2P file-sharing networks are popular means for exchanging files. File-sharing networks have been widely deployed, allowing users to search content, such as songs and videos. In these networks, requests for content trigger, in a transparent way, a series of protocols for content location and transfer. End-users are unaware of all underlay protocols and infrastructure and their main concern is to describe in a proper way the desired content.

However, file-sharing networks suffer from content pollution attacks. Being mainly used for illegally exchanging protected intellectual property products, file-sharing networks cause a big income loss to the content industry, which, in order to protect its products, pollutes these networks. Moreover, being used by thousands of users, file-sharing networks are usually the playground of virus and worms developers. In fact, Liang et al. [72] found out that in the Kazaa file sharing network there existed more than 20.000 versions of some popular files and more than 50%

of them were polluted. The polluted versions in some cases corresponded to more than 60% of the total number of copies of the file. A similar research [73] showed that in the FastTrack filesharing network, for some popular items, about 70% of their copies and 60% of their versions were polluted. Kalafut et al. [74] measured that 68% of all responses in Limewire containing archives and executables, were actually malware. Shin et al. [75] reported that in the KaZaA network, in response to 24 common query strings, over 15% of the results were infected by 52 different viruses.

In this section we present a light-weight solution for ranking information, eventually leading to the isolation of polluted pieces of information. This solution is based on the inforanking algorithm presented in Section 4.2.1.

5.1.1 Design

Reference Architecture

We consider a P2P system in which users initially search for the piece of content they want to download using keywords via the P2P application interface. The P2P application may return thousands of results matching these keywords, especially if the content is popular. Among these results, there will be different versions of the same content. For example the song "The Scientist" may have an mp3 version, a wma version, a version performed by the band "ColdPlay," a version performed by some unknown singer etc. Versions are distinguished from each other by their "meta-data" (e.g., file name, file extension, file properties, keywords etc.). Versions of which the meta-data match their actual content are referred to as *clean versions*, otherwise they are called *fake versions*. Example of fake versions can be an executable file masked as a video file, or a file of a song performed by an artist A, whose meta-data denotes that it is a song of artist B. Each version has multiple copies. All copies of a version are expected to have the same–well known–hash value. A file advertised as a copy of a version is a *clean copy* if its computed hash value matches the expected hash value, otherwise it is a *corrupted copy*.

Why an Information-Centric Approach?

It can be proved that end-users rating is not the best solution in terms of malicious information isolation. This happens due to the fact that users change behavior and a trustful user may suddenly start behaving maliciously. In order to demonstrate that, we simulate a small P2P file-sharing network where end-hosts are rated using a variation of the EigenTrust [76] algorithm. In this network there exist 10 peers, sharing the same 10 files. 50% of them, i.e., 5 peers, are malicious and (when requested) for the 10^{th} file they send bogus data. However, the malicious peers behave in a normal way when they are requested any other item. 100 users enter the network and they start requesting all files in a random order. Every time they receive a file they vote positively or negatively for the file provider. A positive vote is interpreted as the result of a satisfactory transaction, whilst a negative vote as an unsatisfactory one. We assume that voters never lie and every vote is stored safely in a centralized storage accessible by everybody. Thus, every vote is treated by a user as being the result of a personal transaction. As a result we modify EigenTrust as follows: The local trust value of peer i toward j is calculated using the following formula $s_{ij} = sat(*, j) - unsat(*, j)$ where $(un)sat(*.j)$ are all the (un)satisfactory votes for j. The local trust value is normalized as follows $c_{ij} = max(s_{ij}, 0)/max(\sum_j s_{ij}, 0)$. Because of our assumptions there is no need for local trust values transition and aggregation as local trust values are calculated using everybody's (real) votes. Figure 5.1 shows the percentage of bad downloads for queries concerning the 10^{th} file. As it can be seen, user rating did not manage to prevent bad downloads as their percentage is above 40% of the total downloads concerning this specific item. The fact that generally well-behaved users can spread bad content when suddenly they start misbehaving may drive malicious users to start attacking them in order to manipulate them.

Voting

A user's vote for a specific file shows that the user believes that this is a "good" file. Because in inforanking users vote only positively, the fact that a user shares a file can be considered as a vote. Therefore, not only there is no need for

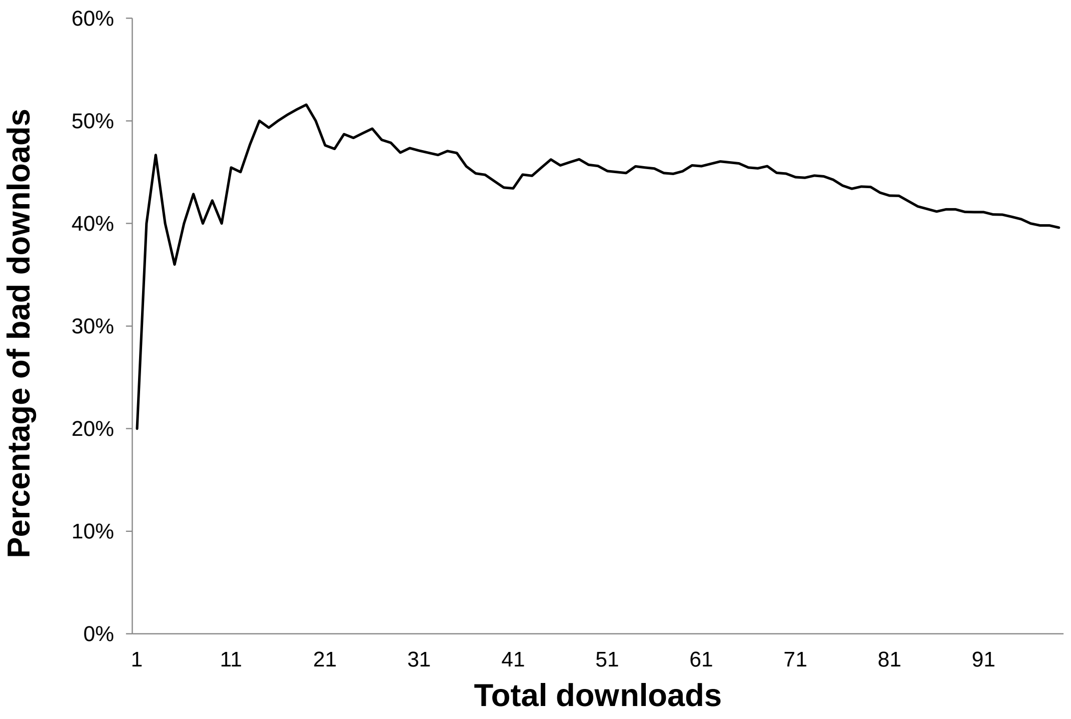


Figure 5.1: Number of bad downloads when user rating is used.

the deployment of a separate voting subsystem, but as long as a user shares a file he participates in the voting procedure, and voting incentives are unnecessary. In a subset V of all files in the system – such as a list of files matching a keywords based search – the score of each file is the sum of all the weighted votes in that subset. Each vote of a user U in V is weighted by a factor w computed as $w = 1/(\sum U_C)^a$ where $\sum U_C$ is the sum of U's votes in V and a is a fixed value. As an example consider a P2P file-sharing network where a user searches for the movie "The free movie". We consider the setup of Figure 5.2 where 7 users in total have versions of that file. Therefore, the user receives the 4 results shown[1] in Table 1. The inforanking based score is calculated here using $a = 1$. The first column of Table 1 contains all the versions that are included in the result set. The second column contains a list of users that share each version and the third column contains the score of each version. As it can be seen in this table, user U1 shares 4 versions in the result set, therefore, he has "voted" 4 times and his vote is weighted by 0.25.

[1]The grouping of files in Versions is based on their hash. If two files have different file name but the same hash, they are considered to belong to the same version.

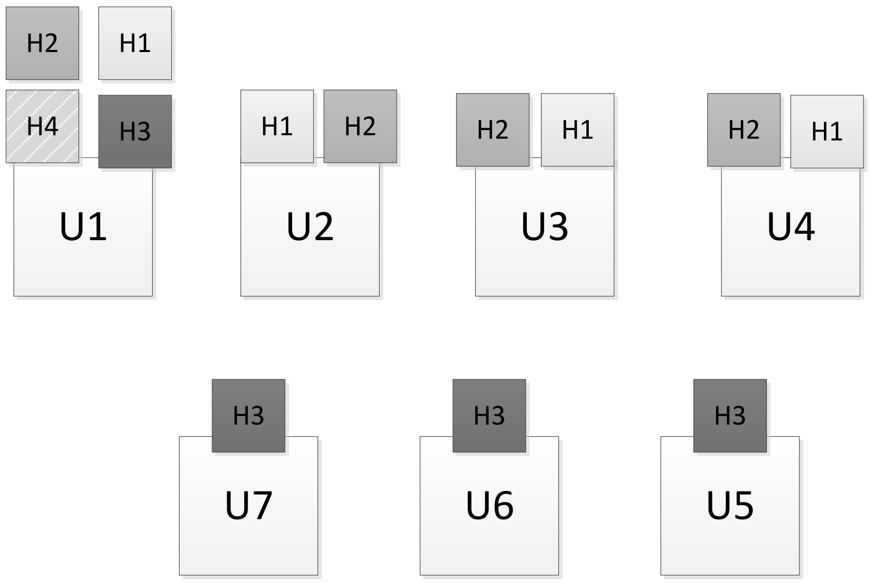


Figure 5.2: Overview of users that share versions of a specific file. In this instance there are 7 users, sharing in total 4 versions of the file.

On the other hand users U5, U6, and U7 have voted only once so their votes are weighted by 1. The rank of each file is calculated by summing the weighted votes. In this example the version with hash value “H3” has the highest score, therefore, this is the one that will be chosen by the user[2]

The score of each version is computed in a distributed way. Upon receiving a result set each user calculates the score of each version using the aforementioned formula. Each user may also maintain a blacklist of versions and users; a user will never download a version contained in the blacklist and will never consider the vote of a blacklisted user.

It can be seen that when inforanking is used, in order for malicious users to achieve a successful attack, they should outnumber legitimate users. In a result set in which each malicious user shares on average F_m versions, each legitimate user

[2]Providing that the user is interested in this version of the movie since inforanking does not take into account user preferences. It is up to the user and the system to isolate files that do not match user preferences.

Versions	**Users**	**Score**
Hash: H1	U1, U2, U3, U4	0.25 + 0.5 + 0.5 + 0.5 = 1.75
Hash: H2	U1, U2, U3, U4	0.25 + 0.5 + 0.5 + 0.5 = 1.75
Hash: H3	U1, U5, U6, U7	0.25 + 1 + 1 + 1 = 3.25
Hash: H4	U1	0.25

Table 5.1: Voting example in a P2P file-sharing network.

shares on average F_g versions and there exist U_g legitimate users, the number of malicious users should be $U_m > (F_m/F_g)^a U_g$ in order to lead to the selection of a fake version. I.e., a significant number of malicious users have to cooperate in order to achieve a successful attack. Moreover, by taking into consideration that fake versions can be blacklisted – either locally, or centrally – malicious users should share multiple fake versions in order to achieve their goal (this is also observed in measurements in real file-sharing systems e.g., in [72, 77]).

By ranking the versions of content rather than the users, inforanking is more robust to attacks where users change behavior, compared to a user-ranking system. Nevertheless, there can be cases in which the number of malicious users is so big that a legitimate user may be convinced to download a polluted item. In those ultra-polluted networks centralized black lists of polluted items can be used. Those black lists will force malicious users to share even more closely related files in order to achieve their attack and, therefore, to lower their vote's weight in inforanking.

Finally, by allowing positive votes only and by considering the fact that a user shares a file as a positive vote, there is no way for attackers to negatively affect the rank of a file.

5.1.2 Evaluation

We evaluate inforanking through the simulation of a high polluted environment and we compare it against a naive solution as well as against the Credence [14] object-based reputation system. We focus on malicious file isolation. We do not consider network or storage overhead since it is negligible in our proposed solution.

The Credence Object-Based Reputation System

Credence is a weighted voting protocol in which a peer may vote positively ($+1$) or negatively (-1) on any file regarding its authenticity. A positive vote is interpreted as an indication that the file's description matches the file's content whereas a negative vote indicates the opposite, i.e, the file's description and its content differ.

The Credence voting protocol works as follows. Any peer wishing to download some content issues a vote-gather query to collect votes on candidate files. This query is flooded to the network and each peer that possesses votes responses. The collected votes are weighted using a weighting factor r. This factor reflects the relationship between two peers and it takes values in the range $[-1, 1]$.

The relationship between two peers, A and B, is expressed by the coefficient of correlation of their voting histories and it is computed as $\theta = (p - ab)/\sqrt{a(1-a)b(1-b)}$ where a and b are the fraction where A and B voted positively, respectively, and p the fraction where both A and B voted positively. The vote weighting factor r equals to θ when $|\theta| >= 0.5$, i.e, when the two peers tend to (dis)agree on the objects they vote positively, and to 0 when $|\theta| < 0.5$,i.e., when the two peers have uncorrelated voting history. For peers which have voted only positively or negatively θ is undefined. In that case a vote agreement metric is used with maximum $|r| = 0.75$.

Each peer maintains a local vote database where the gathered votes are stored. These databases is used to answer incoming vote-gather queries as well as to calculate correlations with other peers. Moreover each peer maintains a list of peers with which it is highly correlated. This list is periodically exchanged between highly correlated peers so as to create transitive correlations. Transitive correlation reflects the notion that if A and B are highly correlated and B and C are also highly correlated then there should also exist a correlation between A and C which is calculated as $\theta_{AC} = \theta_{AB} * \theta_{BC}$

Simulation Setup

Using OMNeT++ and the OverSim framework, we simulate a network consisting of 100 users, 30 of which are malicious. Moreover we consider 10 files with 20 versions each. 12 out of the 20 versions are polluted, containing malicious content. In each simulation round there exists a warming up period during which users select the items that will provide. Each malicious user selects 20 items, while each other user selects 5 items. When all users have selected their items, the non-malicious ones start querying the network in order to obtain a valid version of all the other files they do not have. Every time a user downloads a bad version in the next round it retries to download a valid version of the same file. We simulate and evaluate 3 different strategies. All non-malicious users follow the same strategy. The first strategy is the *naive* strategy. When using naive strategy, users download the version with the most positive votes. If the downloaded file is a valid one, the user gives a positive vote otherwise he votes negatively. The second strategy is the Credence object-based reputation system and the third strategy is inforanking. We also assume that when using the Credence strategy users always vote, and they vote correctly, malicious users vote negatively for valid objects and when θ is undefined $r = 0.5$. Moreover when our solution is used, non-malicious users never share a polluted version. Each simulation round lasts until all non-malicious users download all the valid file. Moreover each experiment is repeated 5 times.

Figure 5.3 shows that even in this highly polluted environment our approach converges much faster than the other two solutions. It can been easily shown that when malicious users share in average V_m versions and non-malicious V_g versions of a file, and there exist U_g non-malicious users in the network, the number of malicious users should be $U_m = (V_m/V_g)^a * U_g + 1$ in order to achieve 1 download of malicious content. Moreover in order for malicious users to be successful and avoid blacklisting it should be $V_m >> V_g$, therefore it is difficult for malicious users

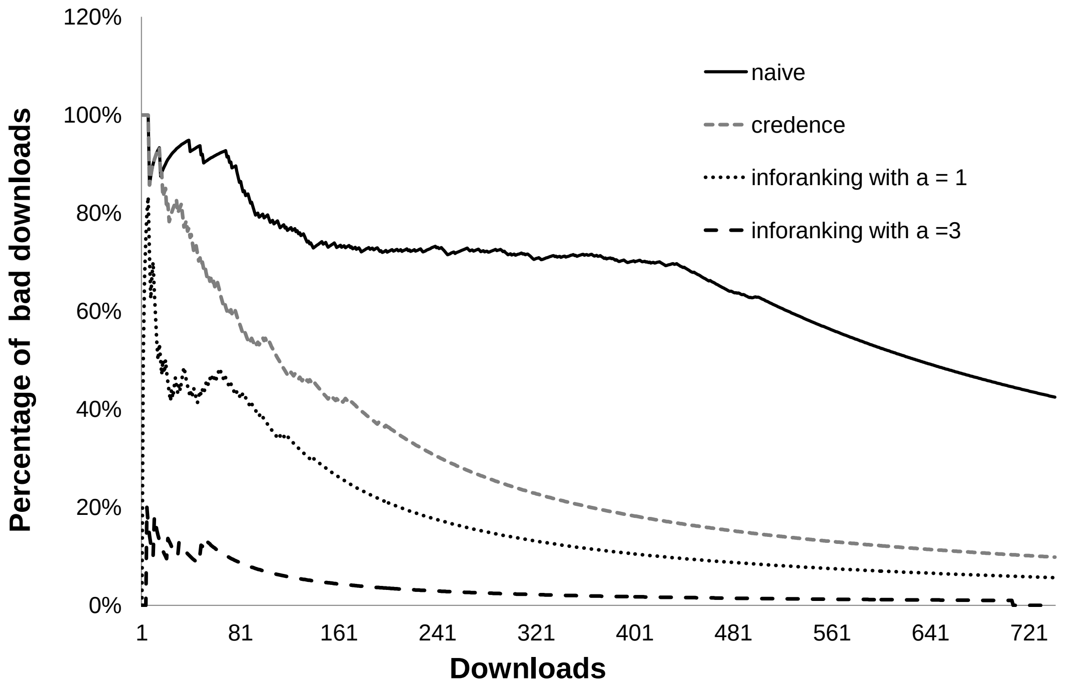


Figure 5.3: Percentage of bad downloads.

to achieve their goal when inforanking is used.

5.2 Access control in Cloud environments

Cloud computing is an emerging paradigm that offers a cost-effective way for outsourcing data storage and computation. Nevertheless, despite its intriguing properties, enterprises are reluctant to fully adopt it, since they are concerned – among other things – about *losing the governance* of their outsourced assets, i.e., losing the ability to enforce their own, enterprise-specific, security policies. According to PwC's Global State of Information Security Survey 2012 [78], the largest perceived Cloud security risk is the "uncertain ability to enforce provider security policies," whereas according to the survey of Subashini and Kavitha [79] one of the biggest security challenges for providing Cloud-based services is the "adherence of the Cloud provider to the security policies of its clients," as well as "the administration of user authorization systems". It is therefore observed that, not only the mismatch between provider-enterprise security policies impedes

Cloud adoption, but overcoming this problem is a challenging task that has to be researched. Indeed, "effective models for managing and enforcing data access policies, regardless of whether the data is stored in the Cloud or cached locally on client devices" was identified back in 2010 as a top research priority, by the European Network and Information Security Agency (ENISA) [80].

One question that may arise is how likely it is for *loss of governance* of the outsourced data to occur, and what is its impact. According to ENISA's Cloud Computing Security Risk Assessment report [81], the loss of governance is a risk with *very high* probability, and with *very high* impact. The same report states that two of the vulnerabilities that may expose an enterprise to that risk are "unclear roles and responsibilities" and "poor enforcement of role definition". This outcome comes as no surprise, since the authentication and authorization systems of the Cloud providers cannot capture the organizational structure and the security policies of each individual enterprise. The interoperability between the corresponding systems requires the development of complex API's which however, according to the Cloud Security Alliance [82], increases the chances of a security breach due to implementation errors. Two notable examples of this case are, the breaking of the SAML-based – a generic XML language used for security assessments between different entities – single sing-on system of Google apps, demonstrated by Armando et al. [83], and the exploitation of implementation errors in multiple SAML systems that allowed an attacker to hijack any user account, demonstrated by Somorovsky et al. [84]. Moreover even if the developed API is implemented correctly, it will be Cloud provider specific, hindering thus the migration of an enterprise to another Cloud provider; this condition is known as *lock-in*, and has been identified as a *high* probability risk by ENISA [81].

Many research efforts have been devoted to enhancing the access control capabilities of Cloud-storage providers, but these solutions either jeopardize user privacy, or introduce significant overhead to the Cloud providers. To this end, we develop a solution that gives full control of the access control assessment to the data owner, introducing minimal overhead for the Cloud provider. This system is based on the access control solution presented in Section 4.1.

5.2.1 Design

Scheme overview

Four basic roles are considered in this solution: the *data owner (owner)*, the *data consumer (consumer)*, the *Cloud provider (CP)*, and the *access control provider (ACP)*. The goal of an *owner* is to store some data in a *CP* and allow *authorized consumers* to perform operations over this data. The data is protected using an *access control policy.* An access control policy is regarded as a function executed in an *ACP*. This function accepts as input a consumer's identification data and outputs either an error message if the user cannot be *authorized*, or an integer number that denotes the access *level* of the consumer. The access level of a *consumer* indicates which operations she can perform over the data that is protected by the corresponding access control policy.

In our scheme, the following trust relationships are considered: the *owner* trusts the *ACP* to authorize a *consumer*, and the *owner* and the *consumer* trust the *CP* to respect the decision of the *ACP*. The first trust relationship type can be trivially established if the *ACP* belongs to the *owner* (e.g., a leveraged enterprise user management system). The second trust relationship is a relaxed form of the currently existing trust relationship between an owner and a Cloud provider: currently, in the best case, an owner trusts a Cloud provider to securely store the owner's business logic, to execute it correctly and to enforce its outcome.

Our goal is to design a system in which the following properties hold:

- *The system is secure*: Provided that all system entities respect the trust relationships described above, it should not be possible for an attacker to perform an operation over some protected data, without being properly authorized.

- *Data consumer privacy is preserved*: In our system a *CP* should gain minimal information about the identity of a *consumer*. Ideally it should only learn that a *consumer* can be authorized by a specific *ACP* and the consumer's *level*. Moreover an *ACP* should not be able to tell the exact data that a *consumer* wants to access.

- *Data can be easily migrated among different Cloud providers*: In our system the only entities that should be aware of the access control policy and its implementation details are the *ACP* and the *owner*. *CPs* are oblivious about the access control policy implementation details. Therefore, providing two *CPs* implement our solution, moving data from one *CP* to another is as trivial as copy-pasting it.

- *An access control policy does not reveal anything about the data and the operations it protects*: In our system an access control policy is decoupled from the data and the operations it protects and it should be defined taking into account solely consumer attributes.

- *An access control policy is re-usable*: In our system it should be possible to use the same access control policy in order to protect many and diverse data items, stored in multiple *CPs*.

- *An access control policy can be easily modified*: In our system the modification of an access control policy should not involve any *CP*; the only entity that should be involved in the modification of an access control policy is the *ACP* where the policy is stored.

A high-level view of the interactions between the system entities is illustrated in Figure 5.4. An execution round of our scheme includes the following steps. Initially an *owner* stores an access control policy in an *ACP* and obtains a *URI* for that policy. As a next step she communicates the obtained URI, as well as the data it protects, to a *CP*, specifying at the same time the required access level(s) for each operation. When a *consumer* tries to perform an operation over some protected data for the first time, she receives as a response from the *CP* a *token* and the URI of the access control policy that protects the data item requested, and she is being redirected to the appropriate *ACP*. Then, the *consumer* authenticates herself to the *ACP*, by providing some form of identification data, and requests authorization, based on the access control policy that corresponds to the obtained URI. The *ACP* checks if the consumer satisfies the stored access control policy; if this is true, the *ACP* *signs* the token, including in the signature

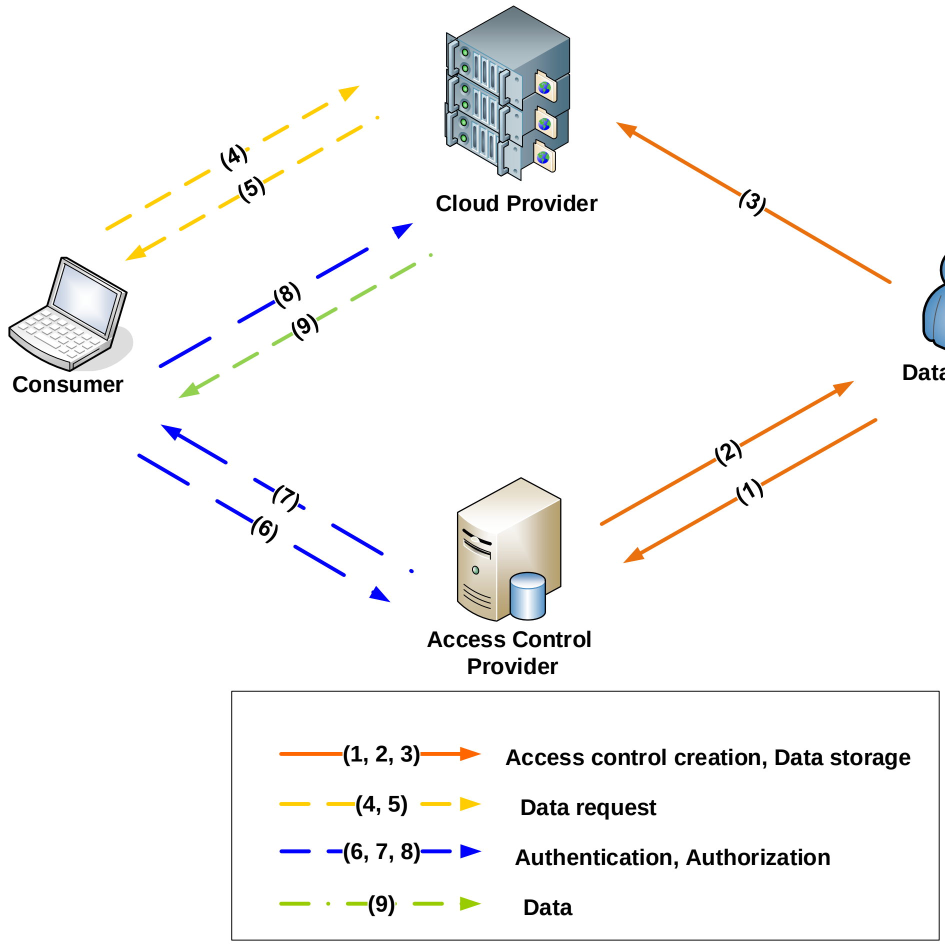


Figure 5.4: Access control delegation for data stored in a Cloud provider.

Table 5.2: Notation

Pub_{CP}	The public key of a CP
Pub_{ACP}	The public key of an ACP
URI_{data}	The URI of a data item stored in a Cloud provider
URI_{acp}	The URI of an access control policy
$Sign_{ACP}(Y)$	The digital signature of plaintext Y using the private key of an ACP

the consumer's access level. The signed token can now be used by the *consumer* in order to perform the desired operation.

Detailed system description

In this section we provide details about our system design. In our system it is assumed that *ACPs* and *CPs* have a pair of public/private keys, and the public keys are known to the *owners*, as well as to the *consumers*. Moreover, it is assumed that all messages are exchanged over a secure channel. Throughout this section the notation of Table 5.2 is used. Our system consists of the following functions :

Access control policy creation and data storage

This function is executed using out-of-band mechanisms. With this function an *owner* initially creates and stores an *access control policy* in an *ACP*. The *ACP* in return provides a URI_{acp}. In order to protect a data item stored in a *CP*, using an access control policy represented by URI_{acp}, the *owner* has to communicate to the *CP* the URI_{acp}, the Pub_{ACP}, as well as the *levels* of *consumers* that are allowed to perform each operation. A *CP* maintains for each data item a URI_{acp}, a Pub_{ACP} and an *Access Table* that contains tuples of the form $< operation, levels >$. A URI_{acp} is re-usable, i.e., it can be used to protect multiple items stored in many *CPs*.

Data operation, unauthorized request:

This function is executed by a *consumer* in order to perform an operation over some protected data, stored in a *CP*. The *consumer* sends a data operation request message to the *CP*. This message contains the operation and a URI_{data}. Upon receiving such a request the *CP* creates a unique *token* and sends it back to the consumer, along with the corresponding URI_{acp}. Therefore, the following exchange of messages takes place:

$$MSG\ \#1:\ Consumer \rightarrow CP : Operation, URI_{data}$$

$$MSG\ \#2:\ CP \rightarrow Consumer : URI_{acp}, Token$$

It should not be possible for a third party to guess a token. In order to keep track of the generated tokens *Cloud providers* maintain a *Token Table* that contains entries of the following form: $< Token, authenticated, expires, URI_{acp}, Level >$. When a new token is generated, a new entry is added to this table, with *authenticated* being set to $false$ and *expires* being set to the generation time plus a very small amount of time, sufficient to obtain an authorization.

Consumer authentication and authorization request:

This function is executed by a *consumer* upon receiving the response of the data operation request. Initially the *consumer* sends her identification data, along with the Pub_{CP} and the URI_{acp} and $Token$ she received with message $MSG\ \#2$, to the ACP responsible for evaluating the access control policy stored in URI_{acp}. The ACP verifies the *consumer's* identification data against URI_{acp}. If the *consumer* satisfies URI_{acp}, the ACP creates a new message that contains the token, the authorization level of the consumer, the amount of time that the token should be valid (i.e., its lifetime), the URI_{acp}, and the Pub_{CP}. Then it signs this message and sends it back to the *consumer*. Therefore during this function the following messages are exchanged:

$$MSG\ \#3:\ Consumer \rightarrow ACP : IDdata, Pub_{CP}, \\ URI_{acp}, Token$$

$$MSG\ \#4:\ ACP \rightarrow Consumer : M2, Sign_{ACP}(M2)$$

where:

$$M2 = Token, Level, Lifetime, URI_{acp}, Pub_{CP}$$

Data operation, authorized request:

With this function a *consumer*, claiming to be authorized, requests to perform an operation over some protected data. The request includes the operation, the URI_{data}, the token, the token's lifetime and the signature of the $M2$ part of the MSG #4 message. Therefore the following message is sent:

$$MSG\#5: \ Consumer \rightarrow CP : operation, URI_{data},$$
$$Token, Level, Lifetime, Sign_{ACP}(M2)$$

Upon receiving this message a CP performs the following actions:

1. Find the token in the Token Table and check if it has expired. If it has expired, return an error

2. If the *authenticated* field of the corresponding record in the Token Table is $false$ then

   (a) Retrieve Pub_{ACP} that corresponds to URI_{data}

   (b) Retrieve the URI_{acp} that corresponds to the token

   (c) Reconstruct the $M2$ part of the MSG #4 message

   (d) Verify $Sign_{ACP}(M2)$, using Pub_{ACP}

   (e) If the signature verification succeeds adjust the expiration time of the token according to the $LifeTime$ field, set *authenticated* equal to $true$, set the appropriate value in the *Level* field, and proceed to Step 3a.

   (f) If the signature verification fails, return an error and exit

3. if the *authenticated* field of the corresponding record in the Token Table is $true$ then

   (a) Find the URI_{acp} and the level that corresponds to the token, from the Token Table

(b) Find the URI_{acp} and the level for the requested operation that corresponds to the URI_{data}, from the Access Table

(c) Check if the retrieved values match. If they match perform the operation, else return an error

Once the CP adjusts the Token Table and marks a token as authenticated, then the consumer does not have to include the $Level, Lifetime, Sign_{ACP}(M2)$ fields in her subsequent requests; the $Token$ is sufficient.

5.2.2 Implementation

As a proof of concept we implemented a secure file storage service using a popular open source Cloud stack, the OpenStack[3]. In particular we leveraged the functionality of the OpenStack component *Swift*, which is used for building object storage systems. The implemented system allows file storage and retrieval, as well as the following operations over the stored files: organizing files in containers, listing the files of a container, copying a file, moving a file and deleting a file. We implemented our communication channels using HTTPS and we pre-configured the *consumer* software with the public keys of the CP and the ACP components.

Swift-based architectures

A Swift-based object storage architecture is composed of two networks: the internal (private) network that consists of *storage nodes*, and the external (public) network that consists of a *proxy* server and (optionally) an *authentication* server. The proxy server accepts HTTP requests and processes them using a Web Server Gateway Interface. The parameters used in each request are encoded as HTTP headers. Each request is pipelined through a number of add-ons, each of which may transform it, forward it, or respond on behalf of the system to the user.

Objects stored in a Swift-based architecture are organized in a three level hierarchy. The topmost level of this hierarchy is the *accounts* level, followed by the *containers* level (second level) and the *objects* level (third level). The accounts

[3]http://www.openstack.org/

level contains user accounts. Each user account is associated with many containers from the containers level. A container is used for organizing objects, therefore a container is associated with many objects from the objects level. An object may be a file or a folder (that contains other objects). Every object within a container is identified by a container-unique name. Each request for an operation over an object contains a URI that denotes the account, the container and the name of the object in question.

Add-on implementation

The *CP* part of our scheme has been implemented as a Swift add-on added in the pipeline of the add-ons processing incoming requests. For each supported operation a user may specify an account-wide URI_{ACP}, a container-wide URI_{ACP}, or an object-wide URI_{ACP}. For each URI_{ACP} the corresponding Pub_{ACP} is provided. When a request is pipelined for the first time through our add-on, the add-on checks if a URI_{ACP} has been set for the object URI specified in the request (or its container, or its account); if this condition is true, the add-on generates a new *token*, using the token generation mechanism provided by Swift, and creates a MSG #2 message as described in Section 5.2.1. The add-on creates a new entry in the *Token Table* that contains the *token*, as well as the corresponding URI_{acp}. The authenticated field of this entry is set to $false$ and the expiration time is set equal to the current time plus 10 sec. Finally the add-on responds with MSG #2 to the *consumer*.

Upon receiving MSG #2, the *consumer* initiates the authentication and the authorization process, which involves the exchange of messages MSG #3 and MSG #4 with the appropriate *ACP*. In our system we implemented a simple *ACP* that authenticates users using a username and a password, and authorizes them using an access control list stored in an SQLite database. With the reception of MSG #4 the *consumer* is ready to perform an authorized request. The first time an authorized request is made, all parameters of message MSG #5 have to be set. In all subsequent requests only the token is sent to the CP.

5.2.3 Evaluation

Security evaluation

It can be easily observed that our system enhances *consumer* privacy. The only information that a *CP* learns about a *consumer* is that he has a trust relationship with a particular ACP, as well as his level. Of course, the latter can be encoded in such a way that it will not reveal any meaningful information. Any other sensitive information is stored in the (trusted) *ACP*. Moreover, regardless of the lifetime of a token, a *consumer* may drop it and request a new one in order to avoid being profiled by a *CP*. Finally an *ACP* does not gain any information about the actual data item that a consumer wants to access: the only information that the *ACP* learns is the public key of the entity that hosts the desired item.

Another security feature of our system is that access control policies can be easily modified. Access control policies are stored in a single point (in the *ACP*) and all protected assets have a *pointer* to that policy; therefore, the modification of an access control policy does not involve communication with the *CP(s)* in which protected data is stored. When an access control policy is changed, all new *consumers* will be authorized using the new policy, whereas all already authorized *consumers* will be re-authorized with the new policy when their token expires.

We now proceed to the security analysis of our system using the threat model proposed by Wang at al. [85], adapted to the context of our system. For our analysis we consider three different attack scenarios: (A) a malicious entity that can be authorized under an access control policy P_{mal}, acting as a consumer trying to perform an operation over a data item protected by an access control policy P_{leg}, with P_{mal} and P_{leg} stored in the same *ACP*, (B) a malicious entity that acts as a *CP* pretending to host an item protected by an access control policy P_{leg}, and trying to access a data item protected by P_{leg} stored in a different CP, and (C) a malicious entity trying to impersonate a consumer from the same system. In all cases we assume that messages are exchanged through a secure channel and communication endpoints cannot lie about their identity. Finally, we do not consider the case in which a malicious entity acts as an *ACP* and steals the credentials of a consumer, since this attack is out of the scope of our system.

Malicious entity acting as a consumer

In this attack scenario a malicious entity, Con_M tries to perform an operation over an item protected by an access control policy P_{leg}, stored in ACP_A. Con_M does not abide by P_{leg}, but he abides by another access control policy, namely P_{mal}, also stored in ACP_A. Con_M's goal is to obtain a $MSG\ \#4$ message in which the $M2$ part would be equal to $(Token, Level, Lifetime, \mathbf{URI_{P_{leg}}}, Pub_{CP})$. Under normal circumstances Con_M will receive a $MSG\ \#4$ message with an $M2$ part of the following form $(Token, Level, Lifetime, \mathbf{URI_{P_{mal}}}, Pub_{CP})$. If Con_M simply replaces $URI_{P_{mal}}$ with $URI_{P_{leg}}$ then $Sign_{ACP}(M2)$ will not be valid anymore, therefore the CP will understand the attack. The only way to include $URI_{P_{leg}}$ in message $MSG\ \#4$, with $Sign_{ACP}(M2)$ being valid, is to include $URI_{P_{leg}}$ in message $MSG\ \#3$, i.e., have Con_M send to ACP_A a message $MSG\ \#3$ of the following form: $IDdata, Pub_{CP}, URI_{P_{leg}}, Token$. However since Con_M does not abide by $URI_{P_{leg}}$ this message will result in an error.

Malicious entity acting as a *CP*

In this attack scenario we assume that the attacker's goal is to perform an operation over a data item $Item_A$ stored in CP_A and protected by an access control policy P_A, stored in ACP_A. The attacker acts a Cloud provider, CP_B, which hosts a data item, $Item_B$, also protected by P_A. Moreover the attacker is able to lure a consumer Con_L, that abides by P_A, to perform an operation over $Item_B$.

The attacker initially sends a message $MSG\ \#1$ to CP_A and obtains a $Token_A$; in order for this attack to be successful the attacker has to obtain a $MSG\ \#4$ message with an $M2$ part of the following form $(\mathbf{Token_A}, Level, Lifetime, URI_{P_A}, \mathbf{Pub_{CP_A}})$. When Con_L requests to perform an operation over $Item_B$, stored in CP_B[4], the attacker responds with a message $MSG\ \#2$ of the following form: $URI_{P_A}, Token_A$. Subsequently Con_L sends a message $MSG\ \#3$ to ACP_A of the following form: $IDdata, Pub_{CP_B}, URI_{P_A}, Token_A$, and receives a message $MSG\ \#4$ with an $M2$ part of $(\mathbf{Token_A}, Level, Lifetime, URI_{P_A}, \mathbf{Pub_{CP_B}})$. In order for the attacker to obtain the desired message he has to replace Pub_{CP_B}, with Pub_{CP_A}, but in this case $Sign_{ACP}(M2)$ will not be valid anymore, therefore CP_A will detect the attack.

[4]According to our assumptions, the attacker cannot pretend to be CP_A.

Malicious entity co-located with a consumer

This attack scenario is applicable when a CP maintains a user management system and associates operations over protected data with particular users (e.g., for charging reasons). In these cases a CP maintains in its Token Table the identifier of the (CP) user for whom the token has been generated. The goal of an attacker in this scenario is to make a CP believe that a consumer Con_L wants to perform an operation OP_A over an item $Item_A$ protected by access control policy P_A. For this scenario it is assumed that the attacker is also a valid CP user and he is eligible to perform OP_A over $Item_A$. Moreover it is assumed that the attacker is able to inject messages on behalf of Con_L.

In order for this attack to take place, the attacker requests to perform OP_A over $Item_A$ and proceeds through all steps until he receives MSG #4. At this point, instead of sending MSG #5 on behalf of himself, he sends it on behalf of Con_L. It can be easily observed that this attack is trivially mitigated since the CP also maintains the identifiers of the users that correspond to each token, therefore MSG #5 will be rejected. It should be noted however that this is possible due to our design choice to have the CP generate the tokens, which is not always the case in other similar systems. This attack, for example, was successfully exploited by Wang at al. [85] against three popular Websites that were using Facebook connect and Twitter OAuth for associating their user accounts with their corresponding Facebook and Twitter accounts.

Overhead

In our implementation, HTTP methods (GET, PUT, DELETE) are used for denoting the desired operation. The size of the RSA keys is 2048 bits and the keys are encoded in JSON format. The size of an encoded key is 400 bytes. Every other field is encoded as a string of hexademical digits: tokens are encoded in a 32 byte string, the digital signatures in a 512 byte string and the token's lifetime in an 8 byte string. Finally, a single byte is used to represent access levels. When a *consumer* wants to perform an operation over some data stored in a CP, protected by an URL_{acp}, a number of messages has to be exchanged. If an ACP has already asserted that the *consumer* abides by the URL_{acp}, and the corresponding authenticated *Token* (that has been generated by the *CP*) has not expired, then a single message from the *consumer* to the *CP* has to be sent. In any other case five messages have to be exchanged: three between the *consumer* and the *CP*, and two between the *consumer* and the *ACP*.

Chapter 6

Discussion and future work

In this Chapter we position our work in the related research fields by comparing our solutions with related approaches. Moreover we discuss deployment issues and we identify future research targets.

6.1 Comparison with related work

6.1.1 Access Control

To our knowledge there is no related research effort addressing the problem of access control enforcement delegation in the context of ICN. Access control issues have been mainly tackled using cryptographic solutions in information naming or at the packet level (e.g., [40]). Nevertheless these solutions simply transfer the problem of access control to the endpoints.

The access control solution proposed in this dissertation has been inspired by single sign-on (SSO) systems–such as OpenID [86] and Shibboleth [87]. Nevertheless our solution differs from SSO in a significant way: SSO systems are based on the so-called *proof-by-possession* primitive, i.e., users authenticate themselves to a RP by providing a token issued by an Identity Provider. This token can be in the form of a web cookie, a HTTP field or a security "ticket". This token however may constitute a security [85] or privacy threat. The secure implementation of this token is complicated and even popular SSO providers – including Facebook and

Google – have been proved vulnerable to severe security attacks [85]. Our solution does not suffer from these problems.

Access control using anonymous credentials – such as in [88, 89] – as well as schemes for delegating user information – such as OAuth [90] – are closer to our work. In these systems the RP evaluates access control policies. In order to perform this task, it is granted access to the necessary user attributes, i.e., the attributes required in order to evaluate an access control policy. In our solution the RP neither gets access to any user information nor does it evaluate any access control policy. The only entity that has access to both user attributes and access control policies is the ACP. This approach has many advantages: it safeguards user credentials, it preserves user privacy and it releases RP from the burden of evaluating access control policies. Moreover, our approach makes easier the creation of access control federations and it is easier to be adopted.

Privacy preserving access control schemes – e.g., [91, 92] - -and decentralized access control mechanisms for cloud services and distributed systems - -e.g., [93, 94] – are orthogonal to our work. Those schemes provide cryptographic primitives that enable outsourcing data storage as well structures that enable the co-operation of various access control mechanisms. Those tools can be used by RPs for securely storing data and by AcPs for creating chains of trust. In any case these mechanisms are transparent to our system, which operates at a higher layer.

6.1.2 Spam protection

Spam in the context of ICN has not been widely studied. To our knowledge, the only related work in this area is by Tarkoma [12], which studies spam in publish-subscribe ICN architectures. Tarkoma proposes an infrastructure-based solution in which each entity digitally signs every message it sends or forwards. Whenever a message is received, the receiver checks whether this message was sent/forwarded by an entity that is considered spammer, by consulting a globally accessible black list of spammers. Our solution differs from Takoma's approach. Instead of trying to isolate spammers we isolate spam information. The reasoning behind this approach is that, if user ranking is used, malicious entities will try to hijack legitimate user identities (using for example viruses and worms). Moreover, if self-certified identifiers are used for content items (e.g., the result of a hash function over the item data) spam items can be easily black listed.

An area where similar phenomena exist is that of P2P file-sharing networks. In these networks malicious users try to distribute "polluted" content. Various content pollution prevention mechanisms have been proposed. These mechanisms can be distinguished into those that rank users and those that rank content. User ranking mechanisms –s uch as EigenTrust [76] and Scrubber [95] – use voting schemes which allow each user to rank others based on their behavior. The voting results are used for building trust relationships. These relationships are propagated, leading to the creation of chains of trust. The rank of each user is calculated in a distributed manner, i.e., each user calculates his own personal ranking of other users. User ranking approaches suffer from two basic drawbacks (i) a newcomer does not know who to trust and (ii) usually it is easy for a user in a P2P system to change his identity, therefore to "reset" his rank. Our approach overcomes these problems: new subscribers do not have to establish trust relationships and the identity of content items usually is constant and unchangeable.

A typical solution for ranking content is Credence [14]. Credence is a weighted voting protocol in which a user may vote positively or negatively on any object regarding its authenticity. Any user wishing to download some content issues a vote-gather query to collect votes on candidate objects; this query is flooded to the network. By using positive only votes, our solution is more lightweight and effective.

The positive only votes approach of our solution has been inspired by PageRank [96]. PageRank is an algorithm for ranking webpages. The incoming links towards a page are considered as positive votes. Each link is weighted in relation to the rank of the page of origin as well as to the number of outgoing links of the page of origin. Our solution borrows the positive only votes approach of PageRank and adapts it for ICN.

6.1.3 User privacy

Onion routing and mix-based mechanisms – such as Tor [97] – are common approaches for protecting user privacy. These mechanisms are based on an overlay network or on intermediate proxy servers, which can be used to route user (subscriber) queries anonymously through circuits. However, such mechanisms do not provide full privacy as queries are eventually revealed to the recipients: if an end-to-end authentication mechanism is used (e.g., only logged on subscribers are allowed to make queries), then subscriber identity is revealed. But even if subscribers are anonymous, the result of their queries can reveal their identity, as it happened with the anonymized query database released by AOL in 2006 [98]. Moreover, circuits introduce latency and hide the subscriber's real location, making it hard to deploy multicast and mobility solutions, affecting the user's quality of experience.

Mechanisms developed for privacy preserving data analysis [99] are not suitable for our case. These mechanisms protect responders privacy by adding permanent fuzziness to the responses, as well, as by hiding their identity using proxies. In our solution, the fuzziness introduced in responses is not permanent: a subscriber can recover the exact information requested. Moreover our solution deliberately does not hide the end-point's identity, which allows for deploying dedicated access control mechanisms.

Private information retrieval (PIR) schemes (see [66] for a survey on these systems) are similar to the technique used in our solution. These schemes are used in order to retrieve a record from a database, without revealing the record or the query. In their basic form, PIR schemes model the database as a large string, or an array, from which bits are retrieved. Variations of PIR schemes use multiple replicas of the same database (e.g., [100]), or split a single information item in many sub-databases (e.g., [101]). Our solution considers a different organization of information: in terms of PIR we consider many individual databases, hierarchically organized, in which the index of a record denotes the path to the database in which it is stored. This organization is very similar to the information space organization of many lookup services.

Information lookup privacy can be regarded as the reverse of searching over encrypted data (e.g., [102]). In these systems, data is encrypted and queries are revealed to the service provider. Nevertheless, even if the provider does not have access to the data, it can infer certain information about subscribers by simply examining their queries, e.g, in a system in which subscribers query for stock prices, the provider will not learn the stock prices, but will learn in which stocks the consumer is interested. Therefore, these systems do not provide query anonymity. Schemes that support searching over encrypted data using encrypted queries – such as [103] – overcome this shortcoming, but they limit the number of subscribers that can perform queries over a set of (encrypted) data items. In these systems queries and data cannot be encrypted using independent keys, therefore the subscribers that perform queries over the same data items, have to share a secret. Our solution does not impose any relationship among the subscribers that lookup the same information items.

Broker-based privacy-preserving schemes presented in [104] and in [71] have similar goals with our solution. However in both approaches queries and content identifiers are encrypted. Moreover in [71] any *subject* may learn some of the keywords included in the *issuer*'s query. In our solution advertisements of information are in plaintext, in order to facilitate the information space management and no entity learns anything about the subscriber preferences.

To the best of our knowledge, privacy preservation, in the context of ICN, has not been widely studied. DiBenedetto et al. [51] have proposed a Tor-like approach, which however suffers from the same limitations as the traditional Tor system. Arianfar et al. [53] introduced a solution that is based on the creation of many algorithmically related identifiers for the same item. These identifiers have the property that they can be easily generated, but given an identifier, an adversary cannot tell to which item it belongs. This solution however adds significant network overhead as all the identifiers have to be advertised. Moreover, a publisher has to perform many computations and permanently store the results, in order to achieve a significant level of privacy.

6.2 Discussion

In Section 2.8 we defined a common ICN model based on which we described our solutions. The main characteristic of this model is the existence of a (logically) unique RP which handles all the subscriptions and advertisements of a specific content item (or of a category of items).

Concepts similar to RPs exist in most ICN architectures discussed in Chapter 2. The DONA architecture uses the Resource Handlers (RH), the PURSUIT architecture uses the Rendezvous Network (RENE), the SAIL architecture uses the Name Resolution System (NRS), the COMET architecture uses the Content Resolution System (CRS), the Convergence architecture uses the Name Resolution System (NRS), and the MobilityFirst architecture uses the Global Name Resolution Service (GNRS). The NDN architecture, however, does not use any similar entity. We discuss how our solutions can be adapted for NDN at the end of this section.

Another aspect that should be taken into consideration when applying our solutions is caching. The access control solution can be applied in caches, however caches should abide by our protocol. When our spam protection solution is used, caches should not respond to subscriptions to scopes: a cache does not have all content advertisements neither knows all user votes, therefore it cannot calculate the score of an item. Finally, when our privacy solution is used, caches are not able to understand subscriber queries.

6.2.1 Considerations for the NDN architecture

The NDN architecture does not include any structure similar to a RP.

Our access control solution can be applied in NDN, if we consider that the RP is “embedded” in the publishers and in the content routers: since all content interests reach a publisher, or a content router that has the content in each data store, this publisher (or content router) can execute the the RP-specific part of the protocol.

Our spam prevention solution requires an entity that collects all the advertisements of similar items. This can achieved using two approaches: (i) a publisher collects, using an application-specific protocol, all similar item advertisements and advertises a content identifier prefix (that is used as the scope identifier), or (ii) “Custodians” [41] are used to advertise a content identifier prefix, which again is used as the scope identifier.

Our privacy solution is the most difficult to adapt for NDN. Since NDN does not have a separate resolution service, our solution can be adapted and used directly on the contents of a publisher. Nevertheless, this would introduced significant computational overhead to the publishers. The adaptation of our solution to NDN-like architectures remains an open research topic.

6.3 Future Work

We now discuss topics for future research for each of the proposed security solutions

Access Control

The proposed solution consideres a single ACP per administrative domain and a single access control policy per content item. Future work in this area includes support for ACP federations and support for multiple policy associations per single content item.

Moreover, we only implemented our solution for the PURSUIT ICN architecture. Our research agenda includes the implementation of our scheme for other ICN architectures. Our solution creates a new business opportunity. We envision that a new market can arise due to our solution, that of the ACPs. Existing security companies can utilize their expertise to offer cutting edge access control services without investing in the ICN (or Cloud market. Moreover, existing social networks may leverage their role to act as ACPs.

Spam

The core of our solution is the inforanking algorithm. We believe that fighting spam is only one of the many possibilities that inforanking may offer. We anticipate that inforanking can be used in recommendation systems, in participatory sensing applications, or even as an anti-DoS mechanism. Moreover we believe that the NDN ICN architecture can also be used as a target for spammers, therefore, our solution could also be implemented for that architecture.

Privacy

Homomorphic encryption is a new, exciting field under active research. By exploiting recent advances in this field we believe that our solution can be expanded into supporting flat RENEs (e.g., DHT based), as well as, rich subscription requests (e.g., requests for a range of items, or requests that specify matching criteria). Moreover, future advances in this field may permit the application of our solution directly on content items.

Chapter 7

Conclusions

ICN has emerged as a promising candidate for the architecture of the Future Internet. Inherently, ICN has many security advantages, however, it also creates new security threats and challenges. To this end, we revisited key ICN proposals, we defined security requirements and we created new security solutions.

The first solution we presented, is an access control delegation scheme that tackles the problem of access control in ICN architectures in an efficient and radically new way. A new trusted entity, the Access Control Provider (ACP), protects subscriber credentials and preserves their privacy. By embedding a pointer to an access control policy – and not the access control policy itself – in a content item, any entity that handles this item can protect it using this policy despite the fact it has no access to the policy definition. We demonstrated the feasibility of our scheme through a proof of concept implementation. Moreover we show the applicability of the proposed solution in the current Internet: We developed a scheme that enables users to outsource data storage and computation to a Cloud provider, without losing governance of their assets. Since Cloud providers are oblivious of the access control policy implementation details and business logic behind it, this scheme relieves Cloud providers from the burden of implementing complex security solutions and facilitates data migration from one Cloud provider to another. We demonstrated our approach through a proof of concept implementation, using a real, publicly available, Cloud stack system.

A possible, not well-studied, threat for ICN is that of spamming. To this end, we developed a light-weight solution for fighting spam in ICN based on the inforanking algorithm. We compared our solution to a publisher ranking based solution and we found that our solution offers better protection against spamming. Moreover our solution uses, to a large extent, functionality already deployed in an ICN architecture and it needs only a few extra messages towards a RN. We also examined the case in which malicious subscribers exist in the system and try to affect it in favor of spammers, and we determined that even in this case our solution is robust enough. Moreover we investigated the possibilities of applying this solution in existing P2P file sharing networks in order to isolate "polluted" files. We found that our solution is able to isolate polluted items without imposing any overhead to the system. We compared our solution to the Credence object reputation system and we proved that our scheme is more effective in terms of how fast a polluted item is identified and isolated.

Finally, we presented a solution for protecting subscriber privacy by offering *unobservability* of their subscriptions. In particular, we developed a PIR protocol that enables a subscriber to learn the RN that has a matching entry for a particular piece of content in a privacy preserving manner. By leveraging the hierarchical organization of the RENE, the proposed solution offers significant improvements, compared to traditional PIR mechanisms, in terms of communication overhead.

Appendix A

Acronyms

AS Autonomous System

ACP Access Control Provider

BN Border Nodes

CaR Content-aware Routers

CCN Content Centric Networking

CDN Content Distribution Network

CMP Content Mediation Plane

COMET COntent Mediator architecture for content-aware nETworks

CONET Content Network

CP Cloud Provider

CR Content Router

CRS Content Resolution System

CS Content Store

CURLING Content-Ubiquitous Resolution and Delivery Infrastructure for Next Generation Services

DHT Distributed Hash Table

DONA Data-Oriented and beyond Network Architecture

DoS Denial of Service

DNS Domain Name System

EEC Elliptic Curve Cryptography

ENISA European Network and Information Security Agency

EU European Union

FIB Forwarding Information Base

FN Forwarding Nodes

GNRS Global Name Resolution Service

GUID Globally Unique Identifier

ICN Information Centric Network

IBE Identity Based Encryption

IN Internal Nodes

IP Internet Protocol

LDAP Lightweight Directory Access Protocol

MTU Maximum Tranfer Unit

NAT Network Address Translation

NDN Named Data Networking

NetInf Network of Information

NRS Name Resolution System

OSN Online Social Network

OSPF Open Shortest Path First

P2P Peer to Peer

PC Path Configurator

PIR Private Information Retrieval

PIT Pending Interest Table

PLA Packet Level Authentication

PSIRP Publish Subscribe Internet Routing Paradigm **PURSUIT** Publish Subscribe Internet Technology

RH Resolution Handlers

RENE REndezvous NEtwork

RN Rendezvous Node

RV Rendezvous Point

SAIL Scalable and Adaptive Internet Solutions

SAML Security Assertion Markup Language

TM	Topology Manager
TTL	Time To Live
TTP	Trusted Third Party
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VDI	Versatile Digital Item

Bibliography

[1] A. Feldmann, "Internet clean-slate design: what and why?" *SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59–64, 2007.

[2] V. Jacobson, "A new way to look at networking," Google Tech Talk, August 2006.

[3] D. Trossen, M. Sarela, and K. Sollins, "Arguments for an information-centric internetworking architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 2, pp. 26–33, 2010.

[4] J. Rexford and C. Dovrolis, "Future Internet architecture: clean-slate versus evolutionary research," *Communications of the ACM*, vol. 53, no. 9, pp. 36–40, September 2010.

[5] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM*, 2007, pp. 181–192.

[6] NSF Named Data Networking project. [Online]. Available: http://www.named-data.net/

[7] FP7 PURSUIT project. [Online]. Available: http://www.fp7-pursuit.eu/PursuitWeb/

[8] FP7 SAIL project. [Online]. Available: http://www.sail-project.eu/

[9] FP7 COMET project. [Online]. Available: http://www.comet-project.org/

[10] FP7 CONVERGENCE project. [Online]. Available: http://www.ict-convergence.eu/

[11] NSF Mobility First project. [Online]. Available: http://mobilityfirst.winlab.rutgers.edu/

[12] S. Tarkoma, "Preventing spam in publish/subscribe," in *Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on*, 2006, pp. 21–21.

[13] J. Bethencourt, D. Song, and B. Waters, “New techniques for private stream searching,” *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 3, pp. 16:1–16:32, Jan. 2009.

[14] K. Walsh and E. G. Sirer, “Fighting peer-to-peer spam and decoys with object reputation,” in *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, ser. P2PECON ’05. New York, NY, USA: ACM, 2005, pp. 138–143. [Online]. Available: http://doi.acm.org/10.1145/1080192.1080204

[15] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, “A survey of information-centric networking research,” *Communications Surveys Tutorials, IEEE*, vol. 16, no. 2, pp. 1024–1049, Second 2014.

[16] Stanford University TRIAD project. [Online]. Available: http://www-dsg.stanford.edu/triad/

[17] Content Centric Networking project. [Online]. Available: http://www.ccnx.org/

[18] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in *ACM CoNEXT*, 2009.

[19] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, “VoCCN: Voice over content-centric networks,” in *ACM ReArch Workshop*, 2009.

[20] D. Smetters and V. Jacobson, “Securing network content,” PARC, Tech. Rep. TR-2009-01, October 2009.

[21] FP7 PSIRP project. [Online]. Available: http://www.psirp.org/

[22] D. Trossen and G. Parisis, “Designing and realizing an information-centric Internet,” *IEEE Communications*, vol. 50, no. 7, pp. 60–67, July 2012.

[23] J. Rajahalme, M. Särelä, K. Visala, and J. Riihijärvi, “On name-based inter-domain routing,” *Computer Networks*, vol. 55, no. 4, pp. 975–986, March 2011.

[24] K. V. Katsaros, N. Fotiou, X. Vasilakos, C. N. Ververidis, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, “On inter-domain name resolution for information-centric networks,” in *NETWORKING 2012*, ser. Lecture Notes in Computer Science, R. Bestak, L. Kencl, L. Li, J. Widmer, and H. Yin, Eds. Springer Berlin Heidelberg, 2012, vol. 7289, pp. 13–26.

[25] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[26] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: line speed publish/subscribe inter-networking," in *ACM SIGCOMM*, 2009, pp. 195–206.

[27] D. Lagutin, "Redesigning Internet-the packet level authentication architecture," Licentiate Thesis, Helsinki University of Technology, Finland, 2008.

[28] FP7 4WARD project. [Online]. Available: http://www.4ward-project.eu/

[29] SAIL Project. (2013, January) SAIL deliverable B.3 (3.3): Final NetInf architecture. [Online]. Available: http://www.sail-project.eu/deliverables/

[30] ——. (2011, July) SAIL deliverable B.1 (3.1): The network of information: Architecture and applications. [Online]. Available: http://www.sail-project.eu/deliverables/

[31] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "Mdht: A hierarchical name resolution service for information-centric networks," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 7–12.

[32] C. Dannewitz, M. DAmbrosio, and V. Vercellone, "Hierarchical DHT-based name resolution for information-centric networks," *Computer Communications*, vol. 36, no. 7, pp. 736–749, April 2013.

[33] G. Garcia, A. Beben, F. J. Ramon, A. Maeso, I. Psaras, G. Pavlou, N. Wang, J. Sliwinski, S. Spirou, S. Soursos, and E. Hadjioannou, "COMET: Content mediator architecture for content-aware networks," in *Future Network & Mobile Summit*, 2011.

[34] W. K. Chai, N. Wang, I. Psaras, G. Pavlou, C. Wang, G. C. de Blas, F. Ramon-Salguero, L. Liang, S. Spirou, A. Beben, and E. Hadjioannou, "CURLING: Content-ubiquitous resolution and delivery infrastructure for next-generation services," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 112–120, March 2011.

[35] COMET Project. (2011, December) COMET deliverable 3.2: Final specification of mechanisms, protocols and algorithms for the content mediation system. [Online]. Available: http://www.comet-project.org/deliverables.html

[36] A. Detti, N. Blefari-Melazzi, S. Salsano, and M. Pomposini, "CONET: A content centric inter-networking architecture," in *ACM Workshop on Information-Centric Networking (ICN)*, 2011.

[37] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi, "Transport-layer issues in information centric networks," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ser. ICN '12. New York, NY, USA: ACM, 2012, pp. 19–24.

[38] A. Baid, T.Vu, and D. Raychaudhuri, "Comparing alternative approaches for networking of named objects in the future Internet," in *IEEE Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, 2012.

[39] T. Vu, A. Baid, Y. Zhang, T. Nguyen, J. Fukuyama, R. Martin, and D. Raychaudhuri, "DMap: A shared hosting scheme for dynamic identifier to locator mappings in the global Internet," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2012, pp. 698–707.

[40] Z. Zhu, J. Burke, L. Zhang, P. Gasti, Y. Lu, and V. Jacobson, "A new approach to securing audio conference tools," in *Proceedings of the 7th Asian Internet Engineering Conference*, ser. AINTEC '11. New York, NY, USA: ACM, 2011, pp. 120–123.

[41] V. Jacobson, R. Braynard, T. Diebert, P. Mahadevan, M. Mosko, N. Briggs, S. Barber, M. Plass, I. Solis, E. Uzun, B.-J. Lee, M.-W. Jang, D. Byun, D. Smetters, and J. Thornton, "Custodian-based information sharing," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 38–43, July 2012.

[42] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (netinf) - an information-centric networking architecture," *Comput. Commun.*, vol. 36, no. 7, pp. 721–735, Apr. 2013.

[43] X. Zhang, K. Chang, H. Xiong, Y. Wen, G. Shi, and G. Wang, "Towards name-based trust and security for content-centric network," in *Network Protocols (ICNP), 2011 19th IEEE International Conference on*, Oct 2011, pp. 1–6.

[44] W. Wong and P. Nikander, "Secure naming in information-centric networks," in *Proceedings of the Re-Architecting the Internet Workshop*, ser. ReARCH '10. New York, NY, USA: ACM, 2010, pp. 12:1–12:6.

[45] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, "Naming in content-oriented architectures," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 1–6.

[46] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named Data Networking (NDN) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[47] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: Seeing the forest for the trees," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, ser. HotNets-X. New York, NY, USA: ACM, 2011, pp. 1:1–1:6. [Online]. Available: http://doi.acm.org/10.1145/2070562.2070563

[48] T. Lauinger, N. Laoutaris, P. Rodriguez, T. Strufe, E. Biersack, and E. Kirda, "Privacy risks in named data networking: What is the cost of performance?" *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 54–57, Sep. 2012.

[49] D. J. Solove, "A taxonomy of privacy," *University of Pennsylvania Law Review*, pp. 477–564, 2006.

[50] A. Pfitzmann and M. Khntopp, "Anonymity, unobservability, and pseudonymity a proposal for terminology," in *Designing Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, H. Federrath, Ed. Springer Berlin Heidelberg, 2001, vol. 2009, pp. 1–9.

[51] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun, "Andana: Anonymous named data networking application," *arXiv preprint arXiv:1112.2205*, 2011.

[52] H.-C. Hsiao, T.-J. Kim, A. Perrig, A. Yamada, S. Nelson, M. Gruteser, and W. Meng, "Lap: Lightweight anonymity and privacy," in *Security and Privacy (SP), 2012 IEEE Symposium on*, May 2012, pp. 506–520.

[53] S. Arianfar, T. Koponen, B. Raghavan, and S. Shenker, "On preserving privacy in content-oriented networks," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking.* ACM, 2011, pp. 19–24.

[54] N. Fotiou, G. Marias, and G. Polyzos, "Towards a secure rendezvous network for future publish/subscribe architectures," in *Future Internet - FIS 2010*, ser. Lecture Notes in Computer Science, A. Berre, A. Gmez-Prez, K. Tutschku, and D. Fensel, Eds. Springer Berlin Heidelberg, 2010, vol. 6369, pp. 49–56.

[55] Trossen, D., ed., "PURSUIT deliverable 2.4, update on the architecture and report on security analysis d(2.4)," April 2013, `http://www.fp7-pursuit.eu/`.

[56] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "Dos ddos in named-data networking," *arXiv preprint arXiv:1208.0952*, 2012.

[57] T. Lauinger, "Security & scalability of content-centric networking," Ph.D. dissertation, TU Darmstadt, 2010.

[58] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*, ser. Lecture Notes in Computer Science, R. Guerraoui, Ed. Springer Berlin Heidelberg, 2001, vol. 2218.

[59] X. Vasilakos, K. Katsaros, and G. Xylomenos, "Cloud computing for global name-resolution in information-centric networks," in *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*, Dec 2012, pp. 88–94.

[60] H. Yuan, T. Song, and P. Crowley, "Scalable ndn forwarding: Concepts, issues and principles," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, July 2012, pp. 1–9.

[61] M. Sarela, C. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott, "Forwarding anomalies in bloom filter-based multicast," in *INFOCOM, 2011 Proceedings IEEE*, April 2011, pp. 2399–2407.

[62] W. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," in *NETWORKING 2012*, ser. Lecture Notes in Computer Science, R. Bestak, L. Kencl, L. Li, J. Widmer, and H. Yin, Eds. Springer Berlin Heidelberg, 2012, vol. 7289, pp. 27–40.

[63] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 2426–2434.

[64] Kjällman, J., ed., "PURSUIT deliverable 3.2, first lifecycle prototype implementation (d3.2)," September 2011, `http://www.fp7-pursuit.eu/`.

[65] N. Fotiou, G. Marias, and G. Polyzos, "Information ranking in content-centric networks," in *Future Network and Mobile Summit, 2010*, 2010, pp. 1–7.

[66] R. Ostrovsky and I. Skeith, WilliamE., "A survey of single-database private information retrieval: Techniques and applications," in *Public Key Cryptography PKC 2007*, ser. Lecture Notes in Computer Science, T. Okamoto and X. Wang, Eds. Springer Berlin Heidelberg, 2007, vol. 4450, pp. 393–411.

[67] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology EUROCRYPT 99*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Springer Berlin / Heidelberg, 1999, vol. 1592, pp. 223–238.

[68] I. Damård and M. Jurik, “A generalisation, a simplificcation and some applications of paillier’s probabilistic public-key system,” in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, K. Kim, Ed. Springer Berlin Heidelberg, 2001, vol. 1992, pp. 119–136.

[69] Y.-C. Chang, “Single database private information retrieval with logarithmic communication,” in *Information Security and Privacy*, ser. Lecture Notes in Computer Science, H. Wang, J. Pieprzyk, and V. Varadharajan, Eds. Springer Berlin / Heidelberg, 2004, vol. 3108, pp. 50–61.

[70] “Advanced crypto software collection,” 2012, http://acsc.cs.utexas.edu/.

[71] A. Shikfa, M. nen, and R. Molva, “Broker-based private matching,” in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, S. Fischer-Hbner and N. Hopper, Eds. Springer Berlin Heidelberg, 2011, vol. 6794, pp. 264–284.

[72] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, “Pollution in p2p file sharing systems,” in *INFOCOM 24th IEEE International Conference on Computer Communications*. IEEE, 2005, pp. 1174–1185.

[73] J. Liang, N. Naoumov, and K. W. Ross, “The index poisoning attack in p2p file sharing systems,” in *INFOCOM 25th IEEE International Conference on Computer Communications*, 2006, pp. 1–12. [Online]. Available: http://dx.doi.org/10.1109/INFOCOM.2006.232

[74] A. Kalafut, A. Acharya, and M. Gupta, “A study of malware in peer-to-peer networks,” in *IMC ’06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 327–332.

[75] S. Shin, J. Jung, and H. Balakrishnan, “Malware prevalence in the kazaa file-sharing network,” in *IMC ’06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 333–338.

[76] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, “The eigentrust algorithm for reputation management in p2p networks,” in *WWW ’03: Proceedings of the 12th international conference on World Wide Web*. New York, NY, USA: ACM, 2003, pp. 640–651.

[77] R. Cuevas, M. Kryczka, A. Cuevas, S. Kaune, C. Guerrero, and R. Rejaie, “Is content publishing in bittorrent altruistic or profit-driven?” in *Proceedings of the 6th International COnference*, ser. Co-NEXT ’10. New York, NY, USA: ACM, 2010, pp. 11:1–11:12.

[78] PwC, “Global state of information security survey,” 2012.

[79] S. Subashini and V. Kavitha, “A survey on security issues in service delivery models of cloud computing,” *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, 2011.

[80] Gorniak, S., ed., “Priorities for research on current and emerging network trends,” *ENISA*, 2010.

[81] Catteddu, D., Hogben, G., ed., “Cloud Computing Benefits, risks and recommendations for information security,” *ENISA*, 2009.

[82] Cloud Security Alliance. (2013) The notorious nine cloud computing top threats in 2013. [Online]. Available: https://cloudsecurityalliance.org/

[83] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra, “Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps,” in *Proceedings of the 6th ACM workshop on Formal methods in security engineering*, ser. FMSE ’08. New York, NY, USA: ACM, 2008, pp. 1–10.

[84] J. Somorovsky, A. Mayer, J. Schwenk, M. Kampmann, and M. Jensen, “On breaking SAML: Be whoever you want to be,” in *Proceedings of the 21st USENIX conference on Security symposium, Security*, vol. 12, 2012, pp. 21–21.

[85] R. Wang, S. Chen, and X. Wang, “Signing me onto your accounts through facebook and google: A traffic-guided security study of commercially deployed single-sign-on web services,” in *Security and Privacy (SP), 2012 IEEE Symposium on*, 2012, pp. 365–379.

[86] D. Recordon and D. Reed, “OpenID 2.0: a platform for user-centric identity management,” in *Proceedings of the second ACM workshop on Digital Identity Management*, ser. DIM ’06, New York, NY, USA, 2006, pp. 11–16.

[87] R. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein, “Federated security: The Shibboleth approach.” *Educause Quarterly*, vol. 27, no. 4, p. 6, 2004.

[88] C. Ardagna, S. De Capitani di Vimercati, G. Neven, S. Paraboschi, F.-S. Preiss, P. Samarati, and M. Verdicchio, “Enabling privacy-preserving credential-based access control with xacml and saml,” in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, June 2010, pp. 1090–1095.

[89] J. Camenisch, S. Mödersheim, G. Neven, F.-S. Preiss, and D. Sommer, “A card requirements language enabling privacy-preserving access control,” in *Proceedings of the 15th ACM symposium on Access control models and technologies*, ser. SACMAT ’10, New York, NY, USA, 2010, pp. 119–128.

[90] E. Hammer-Lahav, “The oauth 1.0 protocol,” 2012, `http://art.tools.ietf.org/html/rfc5849`.

[91] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the 13th ACM conference on Computer and communications security.* ACM, 2006, pp. 89–98.

[92] Q. Ni, E. Bertino, J. Lobo, and S. Calo, “Privacy-aware role-based access control,” *Security Privacy, IEEE*, vol. 7, no. 4, pp. 35 –43, july-aug. 2009.

[93] S. Yu, C. Wang, K. Ren, and W. Lou, “Achieving secure, scalable, and fine-grained data access control in cloud computing,” in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1–9.

[94] S. Miltchev, J. M. Smith, V. Prevelakis, A. Keromytis, and S. Ioannidis, “Decentralized access control in distributed file systems,” *ACM Comput. Surv.*, vol. 40, no. 3, pp. 10:1–10:30, Aug. 2008.

[95] C. Costa and J. Almeida, “Reputation systems for fighting pollution in peer-to-peer file sharing systems,” in *Peer-to-Peer Computing, 2007. P2P 2007. Seventh IEEE International Conference on*, Sept 2007, pp. 53–60.

[96] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web,” Stanford Digital Library Technologies Project, Tech. Rep., 1998.

[97] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” in *Proceedings of the 13th USENIX Security Symposium*, 2004, pp. 303–320.

[98] M. Arrington, “Aol proudly releases massive amounts of private data,” 2006.

[99] C. Dwork, “Differential privacy: A survey of results,” in *Theory and Applications of Models of Computation*, ser. Lecture Notes in Computer Science, M. Agrawal, D. Du, Z. Duan, and A. Li, Eds. Springer Berlin / Heidelberg, 2008, vol. 4978, pp. 1–19.

[100] F. Zhao, Y. Hori, and K. Sakurai, “Two-servers pir based dns query scheme with privacy-preserving,” in *Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on*, 2007, pp. 299–302.

[101] S. Papadopoulos, S. Bakiras, and D. Papadias, “pCloud: A distributed system for practical pir,” *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 1, pp. 115–127, 2012.

[102] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds. Springer Berlin / Heidelberg, 2004, vol. 3027, pp. 506–522.

[103] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith, III., "Public key encryption that allows pir queries," in *Proceedings of the 27th annual international cryptology conference on Advances in cryptology*, ser. CRYPTO'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 50–67.

[104] Y. Xiao, C. Lin, Y. Jiang, X. Chu, and F. Liu, "An efficient privacy-preserving publish-subscribe service scheme for cloud computing," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010, pp. 1–5.