Security Design for an Inter-Domain Publish/Subscribe Architecture

Kari Visala[1], Dmitrij Lagutin[1], and Sasu Tarkoma[2]

[1] Helsinki Institute for Information Technology HIIT /
Aalto University School of Science and Technology, Espoo, Finland
{Kari.Visala, Dmitrij.Lagutin}@hiit.fi
[2] Department of Computer Science, University of Helsinki, Helsinki, Finland
Sasu.Tarkoma@cs.helsinki.fi

Abstract. Several new architectures have been recently proposed to replace the Internet Protocol Suite with a data-centric or publish/subscribe (pub/sub) network layer waist for the Internet. The clean-slate design makes it possible to take into account issues in the current Internet, such as unwanted traffic, from the start. If these new proposals are ever deployed as part of the public Internet as an essential building block of the infrastructure, they must be able to operate in a hostile environment, where a large number of users are assumed to collude against the network and other users. In this paper we present a security design through the network stack for a data-centric pub/sub architecture that achieves availability, information integrity, and allows application-specific security policies while remaining scalable. We analyse the solution and examine the minimal trust assumptions between the stakeholders in the system to guarantee the security properties advertised.

Keywords: Future Internet, publish/subscribe networking, network security

1 Introduction

Data-centric pub/sub as a communication abstraction [2,3,4] reverses the control between the sender and the receiver. Publication in the middle decouples the publisher from the subscriber and there is no direct way of sending a message to a given network, which provides a good starting point against distributed denial of service (DDoS) attacks, where a number of nodes try to flood part of the network with unwanted traffic.

Most pub/sub systems have been overlays on top of IP but our goal is to replace the whole Internet protocol suite with a clean-slate data-centric pub/sub network waist [14]. This enables new ways to secure the architecture in a much more fundamental way compared to overlay solutions, as the underlay cannot anymore be used to launch DDoS attacks against arbitrary nodes in the network. Our architecture comprises of rendezvous, topology, and forwarding functions and the security design presented here covers all these as a whole. In this paper we refine and extend our work in [5] and especially concentrate on the concept of *scope* and how it can be used flexibly to

J. Domingue et al. (Eds.): Future Internet Assembly, LNCS 6656, pp. 167–176, 2011.

support many types of application-specific security policies. Some of the techniques used in our architecture, such as securing of forwarding, delivery tree formation, and rendezvous system interconnection are explained in [5] in more detail.

Our security goals concur with [1] except that confidentiality and privacy are expected to be handled on top of the network layer and are outside the scope of this paper. The security goals are:

- **Availability**, which means that the attackers cannot prevent communication between a legitimate publisher and a subscriber inside a trusted scope.
- **Information integrity**, which guarantees that the binding between the identity of the publication and its content cannot be broken without the subscriber noticing it.
- **Application-specific security policies**, which mean that the architecture can cater for the specialized security policies of different types of applications while partially same resources can be shared by them.

In addition to aforementioned goals, the solution is restricted by the requirements of *scalability* and *efficiency*. For example, it must be assumed that the core routers forward packets at line-speeds of tens of Gigabits per second, which requires expensive, high speed memory for the routing tables. In the inter-domain setting, we have to take into account the various stakeholders such as ISPs, end-users, and governments, and *tussles* [6] between their goals. That is, we cannot enforce certain design choices but keep the architecture flexible and let the balance of power between stakeholders to decide the stable configuration. The design should also adhere to architectural constraints such as the *end-to-end principle* (E2E) [13] by which we mean that the network itself should only have minimal functionality that is required to efficient utilization of the invested resources, and the rest of the features should be implemented on higher layers at the endpoints to be more flexible. The architecture must be incrementally deployable, and minimal in complexity and trust assumptions between stakeholders.

2 Basic Concepts

Data- or content-centric networking can be seen as the inversion of control between the sender and the receiver compared to message passing: Instead of naming sinks to which senders can address messages, the receiver expresses its interest in some identified data that the network then returns when it becomes available taking advantage of multicast and caching [2,3]. We use the term information-centric for this communication pattern to emphasize that the data items can link to other named data and that the data has structure.

An immutable association can be created between a *rendezvous identifier* (Rid) and a data value by a *publisher* and we call this association a *publication*. At some point in time, a *data source* may then publish the publication inside a set of *scopes* that determine the distribution policies such as access control, routing algorithm, reachability, and QoS for the publication and may support transport abstraction specific policies such as replication and persistence for data-centric communication. The

scope must be trusted by the communicating nodes to function as promised and much of the security of our architecture is based on this assumption as we explain in [5]. Scopes are identified with a special type of Rid called *scope identifier* (Sid).

Even though the control plane of our architecture, implementing the rendezvous function, operates solely using data-centric pub/sub model, it can be used to set up communication using any kind of transport abstraction on the data plane *fast path*, that is used for the payload communication. The data-centric paradigm is a natural match with the communication of topology information that needs to be distributed typically to multiple parties and the ubiquitous caching considerably reduces the initial latency for the payload communication as popular operations can be completed locally based on cached data.

Below the control plane, t he network is composed of *domains*, that encapsulate resources such as links, storage space, processing power in routers, and information. The concept of domain is here very general, and can refer to abstractions of any granularity, such as software components, individual nodes, or ASes. An *upgraph* of a node is the set of potential resources, that can be represented as a network map of domains and their connectivity, to which a node has an independent access to based on its location and contracts between providers. We have developed our own language called *advanced network description language* (ANDL) for the communication of network topology information in control plane publications, but it is outside the scope of this paper.

The links in the upgraphs represent resources with minimal abstraction and can be limited to carrying only various *transport* abstractions or protocols over them. Each transport protocol implements a specific communication abstraction and every actualized *instance of interaction* or communication event consuming the resources of the network has an associated transport, *topic*, a *graphlet* and a set of *roles*. For example, when IP is seen as a transport protocol in our network architecture, the roles for the endpoints are a source and a destination or for data-centric transport: a *data source* and a *subscriber*. The topic is identified with an Rid and is used to match the end nodes in correct interaction instances by the scope. For example, for data-centric communication, the topic identifies the requested publication.

A graphlet defines the network resources used for the payload communication and it can be anything from the path of an IP packet to private virtual circuits. Some protocols may require an additional phase for the reservation of a graphlet before the payload communication. A graphlet adheres to a set of scopes that are responsible for policy-compliant matching of nodes to interaction instances, collecting the needed information to build end-to-end paths and placing constraints on the chosen resources for the graphlet. A graphlet connects a set of end nodes that each implement a certain role in the transport.

C ommunication always happens inside at least a single scope, but the policies can be divided into aspects of communication handled modularly by different scopes implemented by different entities. Scopes are responsible for combining upgraph information from multiple nodes requesting to participate in an interaction instance inside the scope [15] and the scope selects a subset of the given resource that adhere to its policies. In cases such as CDNs, the scope can bring its own additional re-

sources to be used in the graphlet. It should be noted, that because a scope can act as a neutral 3rd party for the route selection, it can balance the power between endpoints and optimize the path as a whole.

2.1 Identifier Structure

All identifiers in our system have the similar self-certifying, DONA-like structure [4]. An identifier is a *(P,L)* pair where *P* is the public-key of the *namespace owner* of the identifier and *L* is a variable length label of binary data. Only fixed length hash of the identifier is used in-network, for example in caches, to identify a publication, but variable length names are needed for dynamically generated content, where the data source uses the label as an argument to produce the publication on the fly. This could have been emulated by using only fixed length identifiers, but it would have required additional round-trips in some cases. Because each namespace is managed by its owner, we do not need a special hierarcy or centralized entity managing the labels.

The self-certification is achieved by the namespace owner authorizing a publisher to publish a certain set of labels in the namespace. The actual content segments are then signed by the publisher and the certificate from the namespace owner is included. Together these form a trust chain starting from the namespace owner that can be verified by the subscriber. Here the security model only guarantees the integrity of the association between an identifier and its content. It is assumed that the subscriber knows that she has the correct identifier and trusts the scope enough for availability. For the cases where the content of the publication is known beforehand and no human-readable labels are required, we support a stronger model by allowing the hash of the content to be stored in the label part of the Rid. Confidentiality of publications can be achieved by encryption of the content and/or the labels.

Fig. 1 depicts a simplified example of "My movie edit meta-data" publication that has Rid *(P_N, "My Robin Hood video edit"),* where P_N is the public key of the user's own namespace. The contents of this publication point to another "movie frame data" publication indirectly using a so called *application level identifier* (Aid) of the referred publication. Alternatively, a network level Rid could have been used directly, but they are not assumed to have a long life-time as the security mechanism is coupled with the identifier. We assume that multiple application level schemes for identifiers with different properties will be developed and these can be translated into Rids by various means. DNS is one such orthogonal mechanism that could be used to map long-term human-readable names to Rid namespace public keys.

As can be seen in Fig. 1, the publication contents are split into segments, that each have their own signature chain starting from the P_N verifying the contents of the segment. This makes it possible to check segments independently, for example, before caching. Because it is not feasible to use traditional cryptographic solutions like RSA on a per-segment basis in the payload communication, we use *packet level authentication* (PLA) [25] that uses elliptic curve cryptography (ECC) [23]. An FPGA based hardware accelerator has been developed for PLA [24] accelerating cryptographic operations.

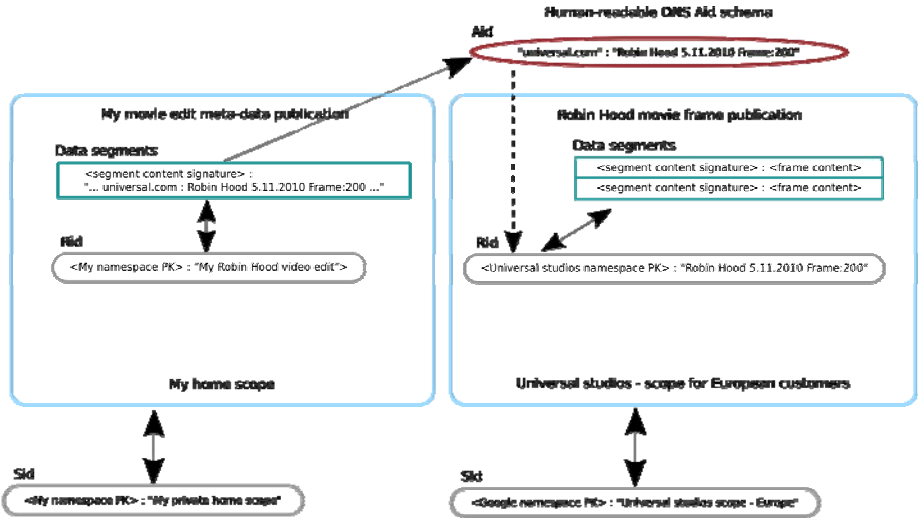
Human-readable DNS Aid schema
Aid
"universal.com" : "Robin Hood 5.11.2010 Frame:200"
My movie edit meta-data publication
Data segments
<segment content signature> :
"... universal.com : Robin Hood 5.11.2010 Frame:200 ..."
Rid
<My namespace PK> : "My Robin Hood video edit">
My home scope
Robin Hood movie frame publication
Data segments
<segment content signature> : <frame content>
<segment content signature> : <frame content>
Rid
<Universal studios namespace PK> : "Robin Hood 5.11.2010 Frame:200"
Universal studios - scope for European customers
Sid
<My namespace PK> : "My private home scope"
Sid
<Google namespace PK> : "Universal studios scope - Europe"

Fig. 1. Publications can refer to other publications persistently using long-term Aids. The scopes, where publications are made available are orthogonal to the structure of the data.

In Fig. 1, the publication on the left is published inside “My home scope” that is fully controlled by the local user. On the other hand, the movie frame publication on the right is stored inside movies studio's localized scope, which can, for example, limit the access to the scope only to the customers of the company. In this example, it is easy to see that the logical structure of the data, e. g. the link between the two publications, is orthogonal to the scoping of the data that determines the communication aspects for each publication.

2.2 Interdomain Structure

Each node has an access to a set of network resources. In the current Internet, most policy compliant paths have the so-called *valley-free* property [16], which means that, on the AS business relationship level, packets first follow 0-n logical customer-provider “up-hill" links, then 0-1 peer-peer links, and finally 0-n provider-customer „down-hill" links. The relationships between ASes are typically based on simple bilateral contracts and we assume that this remains to be the case for the future.

In NIRA [17], it was discovered that almost all policy-compliant paths can be constructed by joining the upgraphs of the communicating endpoints. An upgraph contains transitively all ASes reachable from the node following only customer-provider links and possibly a single peer-peer link as the last hop of the path. The upgraphs contain typically little less than 300 ASes without peering links and around 1500 ASes when taking the peering links into account. The total number of ASes is about 30000, which means that the “two-sided" approach can reduce 50-fold the number of links to consider. NIRA cannot express all BGP policies as the upgraphs will always be the same independent of the packet destination, but our ANDL can be also used to model BGP.

3 Architecture

A central component in our architecture is the rendezvous system, which implements a data-centric pub/sub primitive as a recursive, hierarchical structure, which first joins node local rendezvous implementations into *rendezvous networks* (RN) and then RNs into a global *rendezvous interconnect* (RI) using a hierarchical Chord DHT [18,19] as shown in Fig. 2. The RNs can be implemented, for example, using DONA [4] implementations. In another dimension, the rendezvous system is split into common *rendezvous core* and scope-specific implementations of *scope home* nodes that implement the functionality for a set of scopes.

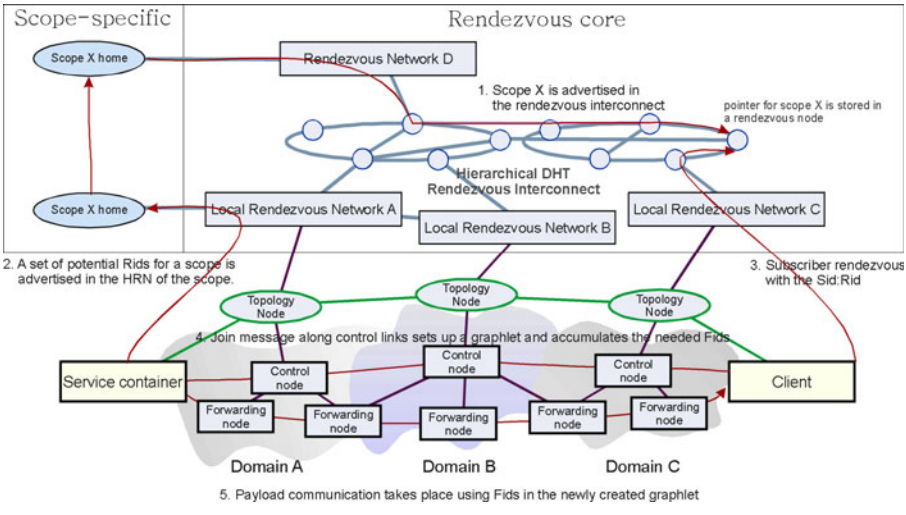

Fig. 2. A client and a service rendezvous on the control plane at the scope home of the scope in which the communication takes place. The scope joins the upgraphs and produces an end-to-end path between the service container (e.g. a data source) and the client (e.g. a subscriber) and returns the information to the client that can then use this information to join a graphlet (e.g. a delivery tree) that can then be used for the fast-path payload communication.

At every level of the hierarchy, the rendezvous core provides an anycast routing to the approximately closest scope home that hosts a given scope. A *subscription message* contains the *(Sid,Rid)* tuple of the publication of which contents the client wishes to receive. Each node in the rendezvous core can cache results and immediately route the answer back along the reverse route to the client. The rendezvous message is first hierarchically routed towards a *scope pointer* based on the hash of the Sid and then the pointer redirects the request towards the approximately closest scope home. Also scope pointers can be cached. *Publication messages* are routed similarly and they contain also the contents of the publication in addition to the *(Sid,Rid)* pair. The scope then stores the contents of the publication so that it can serve the subscribers requesting it. In accordance with the *fate-sharing principle* [20], both the publication and subscription messages need to be periodically repeated in order to keep the publica-

tion data or pending subscription alive. This pub/sub primitive is the only functionality implemented by the rendezvous core. We refer to our work in [5] for a detailed description of the rendezvous security mechanisms.

Scopes, however, can have varying implementations. When a cached result cannot be found in the rendezvous core, the subscription reaches the scope, which can then dynamically generate the response if it has enough information available. It is possible to avoid caching by including version information in the Rid. Each scope implementation may be scaled up by adding more scope home nodes and implementing their own coordination protocol internally. We note that the slow-path control plane rendezvous is not meant for transfer of large publications, but this type of applications should be supported by adding a data-centric transport to the data plane as we did in [2].

Topology manager (TM) is another function that is implemented by each independently managed domain. Its task is to crawl and collect neighbourhood ANDL map and metadata publications of network components using the rendezvous. From this information TM builds a complete view of the local network and publishes this information back into the rendezvous for other nodes to listen to. TM may also hide parts of the network and setup higher-level links in the network map. TM also subscribes to route advertisements from neighbouring domains and updates its upgraph publication accordingly. TM can also act as the local scope home and control its policies. Information about other networks can be found because the publications are named in a standard fashion based on the cryptographic identity of each domain and scope. Basically, a domain X is assumed to have a cryptographic identity DK_X that can be carried in network description attributes. Each domain has a scope with Sid $(DK_X, \text{“external scope”})$ and another scope called “local scope”, which is reachable only locally. Each scope also publishes a meta-data publication inside itself named $(DK_X, \text{“scope meta-data”})$ describing which transports the scope supports, among others. It should be noted that the upgraph combination based routing does not require any type of central entity to manage addresses but the internetwork can freely evolve based on trust between neighbouring domains.

4 Phases of Communication

Each node wishing to communicate first requests the description of end-to-end fast-path resources used for the graphlet from the scope by subscribing to a publication labeled $\text{“<Topic\_Rid>, <}U_N\text{>, } t\text{”}$ where U_N is the *(Sid,Rid)* pair naming the ANDL upgraph map of the node and t is the current time. If the scope performing the upgraph joining is access controlled, then the subscribed label also includes the node identity. The upgraph data itself is published by the provider domain of the node. Because many nodes share the same upgraph, the data-centric rendezvous system caches them orthogonally close to the *scope homes* that are nodes implementing the scope in question. Similarly, the result of the rendezvous is automatically cached and reused if another node in the same domain requests the same service. ANDL maps

can also link to other maps and a complete map can be built recursively from smaller publications, which minimizes the amount of communication as only relevant information needs to be transfered. Multiple scopes can be used together by performing the rendezvous independently for each of the scopes and then taking the logical *AND* of the resulting policy-compliant resources at the end nodes.

A typical sequence of operations is shown in Fig. 2. After a successful rendezvous in the scope, the client is returned information about end-to-end resources that it can use for the graphlet formation. If the transport in question is multicast data dissemination, then a separate resource allocation protocol could be coupled with the protocol as we did in [2]. The client side implementation of the transport would then take the resource description from rendezvous as an input and exchange packets with the selected fast-path domains that would reserve the graphlet. The resource allocation layer could produce, for example, a set of *forwarding identifiers* (Fid), such as zFilter [21], that could be used as an opaque capability to access the graphlet securely without affecting the rest of the network. For this we use the zFormation technique described in [22].

We note that, for example, *onion routing* could be used for the resource allocation, which would hide the destination of the communication from the transit domains and thus guarantee some level of network neutrality. Also pseudonyms for domains can be used to hide the location of the service from its users. We refer to our work in [2] for a more detailed example of graphlet formation in an intra-domain architecture where the dedicated nodes handling a transport can be scattered in the network. The TM of the domain just routes the transports through compatible nodes while balancing the load to each node. This means that the transport functionality does not even have to operate at backbone line speeds because of demultiplexing the flows to multiple nodes. Thus, we claim that the deployment of new transport functionality in the network to be run at branching points of graphlets can be done scalably.

5 Related Work

This section covers related work for publish/subscribe systems and network layer security solutions. A data-oriented network architecture DONA [4] replaces a traditional DNS-based namespace with self-certifying flat labels, which are derived from cryptographic public keys. DONA names are expressed as a P:L pair, where P is a hash of a principal's public key which owns the data and L is a label. DONA utilizes an IP header extension mechanism to add a DONA header to the IP header, and separate resolution handlers (RHs) are used to resolve P:L pairs in topological locations.

In content-centric networking (CCN) [3] every packet has an unique human-readable name. CCN uses two types of packets. Consumers of data send interest packets to the network, and a nodes possessing the data reply with the corresponding data packet. Since packets are independently named, a separate interest packet must be sent for each required data packet. In CCN data packets are signed by the original publisher allowing independent verification, however interest packet's are not always protected by signatures.

Security issues of the content-based pub/sub system have been explored in [7]. The work proposes secure event types, where the publication's user friendly name is tied to the publisher's cryptographic key.

5.1 Security Mechanisms

Most of existing network layer security proposals utilize hash chains or Merkle trees [8]. Examples of hash chain based solutions include TESLA [9], which is time-based hash chain scheme, and ALPHA [10] that relies on interaction between the sender and receiver. While hash chain approaches are very lightweight, they have several downsides, such as path dependency and complex signaling. Merkle tree based solutions have high bandwidth overhead for large trees, and their performance suffers if packets arrive in out-of-order.

Accountable Internet Protocol (AIP) [11] aims to improve security by providing accountability on the network layer. AIP uses globally self-certifying unique endpoint identifiers (EID) to identify and address the source and the destination of the connection, in addition to normal IP addresses. EIDs contain hashes of host's public keys that are communicating within the network. AIP aims to prevent source address spoofing in the following way: If the router receives a packet from the unknown EID, the router will send a verification message back and the node will reply with a message signed by its private key. Since EID is hash of node's public key, this proves that the node owns a corresponding private key and thus has a right to use the EID.

Identity-based encryption and signature scheme (IBE) [12] allows a label, e.g., the user's e-mail address to be used as user's public key, simplifying the key distribution problem. However, IBE relies on a centralized entity called private key generator (PKG), which knows all private keys of its users.

6 Conclusion and Future Work

In this paper we introduced a data-centric inter-domain pub/sub architecture addressing availability and data integrity. We used the concept of scope to separate the logical structure of linked data from the orthogonal distribution strategies used to determine how the data is communicated in the network. This is still ongoing work and, for example, the ANDL language and quantitative analysis will be covered in our future work.

References

1. Wang, C., Carzaniga, A., Evans, D., Wolf, A.L.: Security issues and requirements for Internet-scale publish-subscribe systems. In: HICSS ’02, Hawaii, USA (2002)
2. Visala, K., Lagutin, D., Tarkoma, S.: LANES: An Inter-Domain Data-Oriented Routing Architecture. In: ReArch’09, Rome, Italy (2009)
3. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M., Briggs, N., Braynard, R.L.: Networking named content. In: ACM CoNEXT 2009, Rome, Italy (2009)
4. Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K.H., Shenker, S., Stoica, I.: A Data-Oriented (and Beyond) Network Architecture. In: ACM SIGCOMM 2007, Kyoto, Japan (2007)

5. Lagutin, D., Visala, K., Zahemszky, A., Burbridge, T., Marias, G.: Roles and Security in a Publish/Subscribe Network Architecture. In: ISCC'10, Riccione, Italy (2010)
6. Clark, D., Wroclawski, J., Sollins, K., Braden, R.: Tussle in Cyberspace: Defining Tomorrow's Internet. IEEE/ACM Transactions on Networking 13(3), 462–475 (2005)
7. Pesonen, L.I., Bacon, J.: Secure event types in contentbased, multi-domain publish/subscribe systems. In: 5th international workshop on Software engineering and middleware, pp. 98–105 (2005)
8. Merkle, R.: Secrecy, authentication, and public key systems. Ph.D. dissertation, Department of Electrical Engineering, Stanford University (1979)
9. Perrig, A., Canetti, R., Tygar, J.D., Song, D.: The Tesla broadcast authentication protocol. Cryptobytes 5(2), 2–13 (2002)
10. Heer, T., Götz, S., Morchon, O.G., Wehrle, K.: Alpha: An adaptive and lightweight protocol for hopbyhop authentication. In: Proceedings of ACM CoNEXT (2008)
11. Andersen, D.G., Balakrishnan, H., Feamster, N., Koponen, T., Moon, D., Shenker, S.: Accountable internet protocol (AIP). In: Proceedings of the ACM SIGCOMM 2008, pp. 339–350 (2007)
12. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Brauer, W. (ed.) CRYPTO 1980. LNCS, vol. 84, pp. 47–53. Springer, Heidelberg (1980)
13. Saltzer, J., Reed, D., Clark, D.: End-to-end arguments in system design. ACM Transactions on Computer Systems 2(4), 277–288 (1984)
14. Lagutin, D., Visala, K., Tarkoma, S.: Publish/Subscribe for Internet: PSIRP Perspective. Valencia FIA book (2010)
15. Tarkoma, S., Antikainen, M.: Canopy: Publish/Subscribe with Upgraph Combination. In: 13th IEEE Global Internet Symposium 2010 (2010)
16. Gao, L.: On Inferring Autonomous System Relationships in the Internet. IEEE/ACM Transactions on Networking 9(6), 733–745 (2001)
17. Yang, X., Clark, D., Berger, A.W.: NIRA: A New Inter-Domain Routing Architecture. IEEE/ACM Trans. Netw. 15(4), 775–788 (2007)
18. Rajahalme, J., Särelä, M., Visala, K., Riihijärvi, J.: Inter-Domain Rendezvous Service Architecture. PSIRP Technical Report TR09-003 (2009)
19. Ganesan, P., Gummadi, K., Garcia-Molina, H.: Canon in G Major: Designing DHTs with Hierarchical Structure. In: ICDCS'04, pp. 263–272. IEEE Computer Society Press, Los Alamitos (2004)
20. Carpenter, B.: rfc1958: Architectural Principles of the Internet. IETF (June 1996)
21. Jokela, P., Zahemszky, A., Esteve, C., Arianfar, S., Nikander, P.: LIPSIN: Line speed Publish/Subscribe Inter-Networking. In: SIGCOMM'09 (2009)
22. Esteve, C., Nikander, P., Särelä, M., Ylitalo, J.: Self-routing Denial-of-Service Resistant Capabilities using In-packet Bloom Filters. In: European Conference on Computer Network Defence, EC2ND (2009)
23. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
24. Forsten, J., Järvinen, K., and Skyttä, J.: Packet level authentication: Hardware subtask final report. Helsinki University of Technology, Tech. Rep (2008), http://www.tcs.hut.fi/Software/PLA/new/doc/PLA_HW_final_report.pdf
25. Lagutin, D.: Securing the Internet with Digital Signatures. Doctoral dissertation, Department of Computer Science and Engineering, Aalto University, School of Science and Technology (2010)