Secure Naming for a Network of Information

Christian Dannewitz
cdannewitz@upb.de

Jovan Golić
jovan.golic@telecomitalia.it

Börje Ohlman
borje.ohlman@ericsson.com

Bengt Ahlgren
bengta@sics.se

***Abstract*—Several projects propose an information-centric approach to the network of the future. Such an approach makes efficient content distribution possible by making information retrieval host-independent and integrating into the network storage for caching information. Requests for particular content can, thus, be satisfied by any host or server holding a copy. The current security model based on host authentication is not applicable in this context. Basic security functionality must instead be attached directly to the data and its naming scheme. A naming scheme to name content and other objects that enables verification of data integrity as well as owner authentication and identification is here presented. The naming scheme is designed for flexibility and extensibility, e.g., to integrate other security properties like access control. At the same time, the naming scheme offers persistent IDs even though the content, content owner and/or owner's organizational structure, or location change. The requirements for the naming scheme and an analysis showing how the proposed scheme fulfills them are presented. Experience with prototyping the naming scheme is also discussed. The naming scheme builds the foundation for a secure information-centric network infrastructure that can also solve some of the main security problems of today's Internet.**

I. Introduction

Several research projects, past and current, e.g., Data-oriented Network Architecture (DONA) [1], Network of Information (NetInf) [2], Publish-Subscribe Internet Routing Paradigm (PSIRP) [3], and Content-Centric Networking (CCN) [4] investigate new network architectures based on an information-centric paradigm. In this paradigm, storage for caching information is part of the basic network infrastructure. The network service is defined in terms of Information Objects (IOs) themselves, for example, Web pages, photos, movies, or text documents. The application programming interface has the style of publish/subscribe, where originators publish IOs and receivers subscribe to IOs without the two parties needing to know about each other. Any host or server in the network holding a copy of an IO can thus deliver it to a requesting client. This paradigm thus enables improved content distribution, which appears to be the prevailing usage of the current Internet.

Security in an information-centric network needs to be implemented differently than in current, host-centric networks. In the latter, most security mechanisms are based on host authentication and then trusting the data that the host delivers. In an information-centric network, host authentication cannot be relied upon, or one of the main advantages of an information-centric network, i.e., benefiting from any available copy, is defeated. Host authentication of a random, untrusted host that happens to have a copy does not establish the needed trust. Instead, the security has to be directly attached to the IOs which can be done via the scheme used to name them.

The research leading to these results has been conducted in the 4WARD project and received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216041.

In this paper, we propose the *NetInf naming scheme*. It builds the foundation for an information-centric security model that integrates security deeply into the network architecture. In this model, trust is based on the information itself. Building on an identifier/locator split, each IO is given a unique identifier (ID) with cryptographic properties. Together with additional metadata, the ID can be used to verify data integrity and several other security properties. In comparison with the security model in today's host-centric networks, this approach minimizes the need for trust in the infrastructure, including the hosts providing the data, the channel, and the Name Resolution Service (NRS) (e.g., Domain Name System (DNS)).

The contributions of this paper include analysis and discussion of the naming-related requirements and properties of an information-centric network architecture and a proposal of the corresponding naming scheme that consistently joins these properties together into an overall architecture. In particular, the proposed naming scheme supports the combination of *name persistence*, *self-certification*, *owner authentication*, and *owner identification* all at once, going beyond existing naming schemes for information-centric architectures. Name persistence is ensured if storage *location*, *content*, content *owner*, or owner's *organizational structure* change, e.g., if an employee changes company or a private user changes its ISP. Self-certification allows one to verify the data integrity without requiring trust in the storage location, for both *static* and *dynamic* content. Owner authentication binds an IO to a particular entity, which may be *anonymous*, and owner identification binds an IO to an entity with a particular physically verifiable identity (e.g., a person or company with a given name).

Furthermore, the NetInf naming scheme is designed for *flexibility* and *extensibility* by supporting multiple types of IDs. The naming scheme relies on proven mechanisms like cryptographic hashing and public-key certificate chains to reduce the risk of vulnerabilities. Note that the naming scheme in DONA [1] has some similar properties, but does not allow multiple ID types.

In Section II, the requirements for the naming scheme are discussed. Section III describes its core functionality and Section IV describes and analyzes its main security properties. Relation to other work is addressed in Section V and the results of prototype evaluation are presented in Section VI.

II. Requirements

The main design goal is to design a naming scheme that fulfills the requirements of an information-centric network architecture (as discussed in detail below) and provides improved security properties based on information-centric mechanisms in comparison with today's Internet.

First of all, the naming scheme has to be generic so that it can name any kind of entity, including static and dynamic IOs, services, network nodes, people, and real world entities like places and objects. This requirement results from the flexible NetInf information model that can represent many different entity types. For the same reason and to adapt to future needs, the naming scheme should be extensible, i.e., it should be able to add new information (e.g., a chunk number for BitTorrent-like protocols) to the naming scheme. The need for such extensions is stressed by today's variety of naming schemes (e.g., Digital Object Identifier (DOI) [5] or PermaLink) added on top of the original Internet architecture that fulfill specialized needs which cannot be met by the common Internet naming schemes, i.e., IP addresses and URLs.

To enable efficient, large scale data dissemination that can make use of any available data copy, IDs have to be location-independent. Thereby, identical (or equivalent) data can be identified by the same ID independently of its storage location and improved data dissemination can then benefit from all available copies. This should be possible without compromising trust in data regardless of its network source. Therefore, *self-certification* is a main requirement. Self-certification ensures the integrity of data and securely binds this data to an ID. More precisely, this property means that any unauthorized change of data with a given ID is detectable. Beforehand, secure retrieval of IDs (e.g., via search, embedded in a Web page as link, etc.) is required to ensure that the user has the right ID in the first place. Secure ID retrieval can be achieved by using recommendations, past experience, and specialized ID authentication services and mechanisms.

Another important requirement of the NetInf naming scheme is *name persistence*, not only with respect to storage location changes as discussed above, but also with respect to changes of owner and/or owner's organizational structure and content changes producing a new version of the information. Information should always be identifiable with the same ID as long as it remains *essentially equivalent*. Spreading of persistent naming schemes like DOI also emphasizes the need for a persistent naming scheme. However, name persistence and self-certification are partly contradictory and achieving both simultaneously for *dynamic* content is not trivial.

From a user's perspective, persistent IDs ensure that links and bookmarks remain valid as long as the respective information exists somewhere in the network, reducing today's problem of "404 - file not found" errors triggered by renamed or moved content. From a content provider's perspective, name persistence simplifies data management as content can, e.g., be moved between folders and different servers as desired. Name persistence with respect to content changes makes it possible to identify different versions of the same information by the same consistent ID. If it is important to differentiate between multiple versions, e.g., of Wiki pages, a dedicated versioning mechanism is required, and version numbers may or may not be included as a special part of the ID. Doing versioning in a distributed, collaborative fashion poses additional requirements, which are not addressed in this paper.

We differentiate between two mechanisms to achieve trust and accountability in NetInf. The first is *owner authentication*, where the owner is recognized as the same entity, who repeatedly acts as the object owner, but may remain *anonymous*. The second is *owner identification*, where the owner is also identified by a physically verifiable identifier, such as a personal name. This separation is important to allow for anonymous publication of content, e.g., to support free speech, while at the same time building up trust in a (potentially anonymous) owner.

III. Naming Scheme

Section III-A introduces the basic concepts of the NetInf naming scheme, with details about the ID structure and corresponding metadata given in Sections III-B and III-C, respectively.

A. Basic concepts

The NetInf naming scheme is designed to securely name a wide variety of objects and IOs including all entities listed in the requirements section. To support this variety and potentially diverse requirements, the NetInf naming scheme enables flexibility and extensibility by supporting different name structures, differentiated via the Type tag in the ID.

In the NetInf naming scheme, any entity/IO is represented by a globally unique ID. Together with the IO's data and metadata, an IO is defined as *IO = (ID, Data, Metadata)*. *Data* contains the main information content of the IO. *Metadata* contains information needed for the security functions of the NetInf naming scheme, e.g., public keys, content hashes, certificates, and a data signature authenticating the content. It also includes non-security-related information, i.e., any attributes associated with the IO, e.g., the location where a picture was taken. Metadata can be an integral part of the IO or can be treated and stored independently, as a separate IO.

In an information-centric network, multiple copies of the same IO typically exist at different locations. In contrast to today's Internet architecture, due to the ID/locator split, those identical IOs have the same ID in NetInf. All IOs under the same ID constitute an *equivalence class*, represented by the common ID and need not be identical replicas of each other.

IOs are manipulated (e.g., generated, modified, registered, and retrieved) by physical entities such as nodes (clients or hosts), persons, and companies. Physical entities able of generating, i.e., creating or modifying IOs are called *owners*.

Several security properties of our naming scheme are based on the fact that ID contains the hash of a public key that is part of a public/secret key pair PK/SK. This PK/SK pair is conceptually bound to the IO itself and not directly to the

owner as in other systems like DONA. In NetInf, the PK/SK pair can be bound to the owner only *indirectly*, via a certificate chain. This is important to note because it enables owner change while keeping persistent IDs. The key pair bound to an IO is thus denoted as PK_{IO}/SK_{IO}.

Making the (hash of the) public key part of ID enables self-certification of *dynamic* content while keeping persistent IDs. Self-certification of *static* content can be achieved by simply including the hash of content in the ID, but this would obviously result in non-persistent IDs for dynamic content. For dynamic content, the public key in the ID can be used to securely bind the hash of content to the ID, by signing it with the corresponding secret key, while not making it part of ID.

The owner's PK as part of the ID inherently provides *owner authentication*. If the public key is bound to the owner's identity (i.e., to its real-world name) via a trusted third party certificate, this also allows *owner identification*. Without this additional certificate, the owner can remain anonymous.

Sometimes, there will be a need to authenticate/identify more than one stakeholder separately, e.g., a song that is created by an artist but published by a record company. The NetInf naming scheme allows authentication and identification of multiple stakeholders for the same object by including additional authentication data in the security metadata.

B. ID Structure

The Netinf naming scheme uses flat IDs mainly because we want the IDs to be self-certifying and persistent. In addition, flat IDs are advantageous when it comes to mobility and they can be allocated without an administrative authority by relying on statistical uniqueness in a large namespace. Although IDs are not hierarchical, they have a specified basic ID structure, *ID = (Type, A, L)*, illustrated in Figure 1.

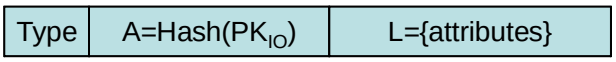

Fig. 1. Basic ID structure.

The *Authenticator* field *A=Hash*(PK_{IO}) binds the ID to a public key PK_{IO}. The hash function *Hash* is a cryptographic hash function, which is required to be one-way and collision-resistant. The hash function serves only to reduce the bit length of PK_{IO}. PK_{IO} is generated in accordance with a chosen public-key cryptosystem. The corresponding secret key SK_{IO} should only be known to a legitimate owner. In consequence, an owner of an IO is defined as any entity who knows SK_{IO} or any other secret key authorized by SK_{IO} via a public-key certificate chain (see Section IV-B).

The *Label* field *L* contains a number of *identifier attributes* associated with an IO. These attributes can be related to the data content and/or the owner or can be serial or random numbers. The pair (*A*, *L*) has to be globally unique. In particular, the identifier attributes:

- provide global uniqueness if PK_{IO} is repeatedly used for different IOs,
- authenticate static IOs or their static parts, by including their hash values,
- enable partially hierarchical name resolution, if the NRS does not treat the ID as flat, e.g., for scalability reasons.

To build a flexible and extensible naming scheme, e.g., to adapt the naming scheme to the named entity type, different types of IDs are supported by the NetInf naming scheme and differentiated via a mandatory and globally standardized *Type* field in each ID. The Type field specifies the hash functions used to generate the ID. If a used hash function becomes insecure, the Type field can be exploited by the NRS in order to automatically mark the IDs using this hash function as invalid. In addition, the Type field also defines the variable format and structure of the Label *L* and how to interpret this structure. In particular, the Type field specifies if the IO is static or dynamic.

C. Security Metadata Structure

The security metadata is extensible and contains all the information required to perform the security functions embedded in the NetInf naming scheme. In particular, the security metadata includes:

- specification of the hash function *h* and the algorithm *DSAlg* used for the digital signature,
- complete PK_{IO} (not only *Hash*(PK_{IO})),
- public-key certificate chain authorizing the signing PK/SK pair (see Section IV),
- specification of the parts of Data and attribute Metadata (if not all) that are self-certified,
- for dynamic IOs, hash of the self-certified data, which, in addition, contains the ID of IO, in order to prevent unauthorized change of the Type and *L* fields,
- signature of the self-certified data signed by SK_{IO} or any other authorized SK,
- if needed, all the data required for owner authentication and identification.

IV. Analysis of Security Properties

The following subsections describe and analyze the main security-related properties: self-certification, name persistence, and owner authentication and identification.

A. Self-Certification

Self-certification relates to all data or just parts of the IO, as specified in the security metadata. If *h* and *DSAlg* are secure, then the only feasible way of performing unauthorized changes to the self-certified data is by changing the IO's ID.

Self-certification of static IOs, whose content does not change in time, can be achieved by including the hash of the self-certified data in the L field of the ID. Verification of the data is then performed by computing the hash value of the retrieved data and comparing it with the one from the ID. More generally, the same holds for any invariant, static parts of the data, whose hash values can be included in the L field of the ID. The advantage of this approach is that there is no need to resort to the metadata in order to verify the content.

For dynamic IOs, the hash of (variable) data cannot be included in the ID as it would violate name persistence. Self-certification is then achieved by storing the signed hash of the self-certified data in the metadata, where the signature is performed by a legitimate owner using SK_{IO} or any other SK authorized by SK_{IO}. Verification of the data then consists of first verifying if the PK used for signing is equal to PK_{IO} (by comparing the hash values) or is authorized by a valid public-key certificate chain originating in PK_{IO}/SK_{IO} and then verifying the signature of the self-certified data. This ensures a secure binding between the self-certified data and the ID. Only the legitimate owners can produce the valid signature and any change to the self-certified data performed by other entities can be detected. If an unauthorized PK/SK pair is used for signing, to produce a new signature for potentially modified data, the ID will change.

B. Name Persistence

The NetInf naming scheme can ensure persistent IDs in spite of changes of the storage *location*, the *content* itself, the *owner* of a data item, as well as owner's *organizational* changes.

Independence of *organizational changes* is an inherent feature of the NetInf naming scheme as IDs, especially the Authenticator field, are flat and do not reflect organizational structures as in other approaches, e.g., CCN.

Location independence results from the ID/locator split that the naming scheme builds upon. The NetInf IDs are *dynamically* bound to one or multiple network locators, where copies of the IO are stored. Hence, when a locator changes, the ID remains persistent and only the binding has to be adapted, which is managed outside the naming scheme by a NRS.

Content independence for *static* content can be achieved by storing the content's hash in the ID as described in Section IV-A. For *dynamic* content, content independence is achieved by storing the signed content's hash in the associated metadata instead of the ID.

Owner independence can be achieved in two ways, by the less complex *basic approach* and the *advanced approach*, which is more secure, but also more complex. The owners can choose the approach more appropriate to their needs.

The *basic approach* is based on the fact that PK_{IO} contained in the ID is bound to the IO itself, and not to a specific owner. Therefore, when the owner changes, the corresponding SK_{IO} can be securely passed on to the new owner, who will then use the same PK_{IO}/SK_{IO} pair. Thereby, the PK_{IO}/SK_{IO} pair is not changed and the ID remains persistent. This approach is simple, but requires a secure (confidential and authenticated) channel for passing on the SK_{IO}. A disadvantage of this approach is that it is not robust with respect to disclosure of the secret key.

In the *advanced approach*, each new owner can use a new key pair PK/SK for self-certification and, possibly, also a new hash function *h* and signing algorithm *DSAlg*. This eliminates the need for securely transferring the secret key and ensures a certain level of robustness with respect to secret key disclosure. To keep the ID persistent, the hash of the original PK_{IO} in the ID remains unchanged. However, the metadata is signed by the secret key of the latest new owner, SK_{latest}, and verified by the corresponding public key, PK_{latest}.

The PK_{latest}/SK_{latest} pair used for signing needs to be securely bound to the ID. This is achieved by using a public-key certificate chain authorizing the PK_{latest}/SK_{latest} pair by the original PK_{IO}/SK_{IO} pair. The public-key certificate chain provides a secure binding between PK_{IO} and PK_{latest} and, hence, also between PK_{IO} and SK_{latest}. Each particular public-key certificate includes the PK of the former owner and the new PK of the new owner. It also contains the IO's ID to bind the authorizations to this ID. The specification of new *h* and *DSAlg'* is also included in the public-key certificate if those have changed. The certificate is signed by the SK of the former owner. To ensure validity of the digital signature, the whole certificate chain, stored in the IO's metadata, needs to be verified, by verifying each certificate along the chain.

Both the approaches technically allow all legitimate owners in the certificate chain to make valid changes to the IO. If this behavior is undesired and former owners should be prevented from making changes to the IO, then the advanced approach facilitates prohibition on a legal basis, by including the production and expiry times in each authorization certificate and by providing a trusted time certification service to the involved owners (e.g., by the NRS during registration/unregistration). Otherwise, a key revocation mechanism can be used for this purpose. The same mechanism can also be used to handle key renovation and revocation of compromised secret keys. Details about this are out of the scope of this paper.

C. Owner Authentication and Identification

A distinctive feature of the NetInf naming framework is that owner authentication is separated from data self-certification. This means that the PK/SK pair used for owner authentication (PK_{owner}/SK_{owner}) is allowed to be different from the one used for data self-certification of an IO (PK'_{IO}/SK'_{IO}), which itself is equal to PK_{IO}/SK_{IO} or is authorized by it.

Owner authentication essentially binds the IO's self-certified content to PK_{owner}. It can be achieved by including $Hash(PK_{owner})$ in self-certified data and by signing this data both by SK'_{IO} and by SK_{owner} (if $SK_{owner} \neq SK'_{IO}$). The signatures are included in the associated security metadata. Owner authentication is then performed by verifying the signatures.

Owner authentication allows the owner to remain anonymous. Nevertheless, by reusing the same PK_{owner} for several IOs, the owner can build up trust in this PK_{owner} and, thereby, in the content itself on the basis of the quality and trustworthiness of the previously published content. Thereby, PK_{owner} becomes a kind of virtual identity of the owner, comparable to, e.g., an eBay user name that has a certain level of trust based on its history of transactions.

Owner identification essentially binds the IO's self-certified content not only to to PK_{owner}, but also to the corresponding real world identity of the owner, e.g., the name of a person or a company. It can be achieved by including the real world

identity in self-certified data and by signing this data in the same way as for owner authentication. In addition, the real world identity needs to be verified, and this can be achieved by an additional signature binding PK_{owner} to this identity, i.e., the public-key certificate. This certificate is issued by a trusted third party upon verifying that the physical entity with this identity knows SK_{owner} and is included in the security metadata. Owner identification is then performed by verifying all the signatures.

V. Related Work

The basic idea that an ID contains an 'object owner'-related part (hash of public key) that can be used for authentication and a 'label' part that is under the control of 'the owner' has been suggested in previous work, which includes Simple Distributed Security Infrastructure (SDSI) [6] and DONA [1]. However, SDSI and DONA bind the public key directly to an entity called *principal*. Therefore, when the principal changes, the ID also changes both in SDSI and DONA, breaking the name persistence. In our naming scheme, we can keep the ID persistent even when the owner changes because the public key is bound to the IO itself and only indirectly to the owner. As a consequence, NetInf also supports anonymous publications, which is difficult to achieve if the public key is directly bound to the owner. Moreover, in DONA, "... only hosts authorized by the principal P can offer to serve (i.e., provide access to) entities with IDs of the form P:L" [1], which limits the usability of available copies. In contrast, it is an important feature in NetInf to make use of any available data copy to improve efficient data dissemination.

The naming approaches of NetInf and CCN [4] differ essentially, because CCN uses hierarchical names typically corresponding to organizational structures. This implies that the name persistence with respect to owner or organizational changes is not satisfied. The CCN security concept requires that the IO's ID and the content be signed by an entity trusted by the users. If this entity is bound to the IO's ID (e.g., the owner or any part of organizational structure), then the trust changes if the owner or organizational structure change. If not, then the trust becomes difficult to control. If this entity is different from the owner, then signing of dynamic content may be a problem. Since the signing public keys are placed outside the ID, CCN IDs do not inherently support self-authenticated name registration and users cannot specify the trusted public keys beforehand when indicating an interest in data and can, thus, be overwhelmed by fake data packets with the 'right' ID, which cannot be filtered on the basis of their ID. This makes the system vulnerable to Denial of Service (DoS) attacks.

Another closely related initiative is PSIRP [3]. Their main idea is to implement a pure publish/subscribe information-centric system. They use rendezvous IDs to retrieve IOs. In addition, they have scope IDs to restrict the distribution of objects. However, to our best knowledge, PSIRP currently only focuses on self-certification via the IDs but not on owner authentication, identification, and other security properties supported by the NetInf naming scheme.

For the special case of *static* content, NetInf can include the cryptographic hash of the underlying information in the ID itself. A similar idea has been proposed, e.g, in SFS [7], however, for self-certifying pathnames that are location dependent. Likewise, the idea of certificate chains (although, in a different context) has been used in other proposals like Simple Public Key Infrastructure (SPKI) [8] and KeyNote [9].

The Handle System [10] offers a general mechanism to persistently identify digital objects and builds the basis for other systems like DOI [5]. However, the Handle system does not include security mechanisms like self-certification and owner authentication in the naming scheme itself, which are required for an information-centric network architecture. In addition, the NetInf naming scheme differs from many of the above mentioned systems because of its flexibility to support various different ID structures via its ID type tag.

During recent years, it has become apparent that many of the problems that are haunting the Internet stem from the semantic overloading of the IP address. A lot of effort has been put into investigating how an ID/locator split can be instrumental in providing better support for mobility, multihoming, protection against DoS attacks, etc. Important work in this area includes the Host Identity Protocol (HIP) [11], the Internet Indirection Infrastructure (I3) [12], the NodeID architecture [13], and the Layered Naming Architecture [14]. Our naming scheme is also based on an ID/locator split, but has different properties as our focus is on supporting efficient data dissemination while at the same time satisfying the requirements of a secure information-centric network architecture.

In the Layered Naming Architecture, the authors state four principles which stress that names should not impose unnecessary restrictions by how they bind to underlying protocols or provide name persistence. To cater for these requirements and allow maximal flexibility, we use flat IDs which allow arbitrary recursions and indirections as they are allowed to map onto themselves before they finally resolve to a locator. Flat IDs are instrumental for the NetInf naming scheme to provide several security properties. Scalable name resolution for large flat namespaces is a challenge. While not being the focus of this paper, the MDHT and LLC architectures [15] give an overview of how this problem is addressed within NetInf.

There are several systems that build their own mechanisms to check data integrity based on cryptographic hashes, including open source software package distribution and the BitTorrent protocol. Such systems can benefit from the NetInf naming scheme, eliminating the need to build their own mechanisms while providing additional security properties.

VI. Evaluation

We have built a Java-based NetInf prototype [16] to evaluate and show the feasibility of the NetInf naming scheme, currently working on FreeBSD, Linux, Windows, and Android. The naming scheme has proven to be easy to implement as it is based on several established security mechanisms like encryption and digital signatures that can be integrated via existing, proven libraries. Likewise, it is also easy to

integrate and use the naming scheme in applications. We have built applications from scratch and have extended existing applications (an email client (Thunderbird) and a Web browser (Firefox)) with additional security functionality based on simple plugins. For example, our Firefox plugin enables the browser to interpret Web pages that contain NetInf IDs instead of regular URLs as links. Thereby, users and content publisher can benefit from all NetInf naming scheme advantages right away by simply using the plugin on the client side and using NetInf IDs instead of (or in addition to, for backward compatibility) URLs as links in Web pages. For example, the plugin gives users an additional icon indicating if the currently received Web page is authentic or has been (maliciously) altered. Publishers, in addition, benefit from more flexible content management thanks to the persistent IDs. More details about the prototype are described in Dannewitz et al. [16]. In general, the naming scheme has demonstrated to improve the security properties of a wide variety of applications ranging from information dissemination and information management to advanced context-aware mobile applications [17].

The generality of the naming scheme allows for several additional use cases. For example, the security features inherent to the NetInf naming scheme enable a secured, *self-authenticated* name registration process of IOs by owners in the NRS, which prevents (replay) DoS attacks on the underlying registration servers that simply replay the transcripts of previous registrations. In order to register/unregister an IO, an owner needs to provide a (fresh) signature of the IO's ID and the registration time, where the signature is verified by using PK_{IO} from the ID. Signing the registration time prevents the replay DoS attacks.

The load and overhead produced by the naming scheme on a client/server node has proven not to be an issue. In fact, we have an implementation of the naming scheme smoothly running on Android cell phones. The overall prototype will be published as open source project shortly.

VII. Conclusion

Information-centric network architectures have an inherent need for a secure naming scheme. Because requested data can be delivered from any available untrusted network location that happens to have a copy of the data, security has to be based on the data and its ID itself and cannot be based on network nodes. There are some existing proposals for information-centric network architectures, including corresponding naming schemes. However, it seems that they are all missing some properties that we think are important.

Having this in mind, we started by investigating what properties are essential for an information-centric naming scheme. Based on this analysis, we have developed the NetInf naming scheme. It can simultaneously fulfill all our requirements based on the combination of a flexible ID structure and a securely attached set of metadata. In addition to security properties like self-certification and owner authentication also provided by some other naming schemes, the NetInf naming scheme is characterized by its unique, non-trivial combination of security-related properties. This includes the flexibility to name a wide variety of entities, extensibility, persistent IDs under various changing conditions (especially owner change), self-authenticated name registration, and support for anonymous publication of information.

Our prototype evaluation shows that our naming scheme is feasible and provides a powerful foundation for a secure information-centric network architecture. In the future, we will extend the naming scheme to support features like advanced version tracking and information-centric access control.

Acknowledgment

The authors thank their colleagues of the 4WARD project [15] for their input and many fruitful discussions.

References

[1] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.

[2] B. Ahlgren, M. D'Ambrosio, C. Dannewitz, M. Marchisio, I. Marsh, B. Ohlman, K. Pentikousis, R. Rembarz, O. Strandberg, and V. Vercellone, "Design considerations for a network of information," in *Proc. Int. Workshop on Re-Architecting the Internet*, Spain, Dec. 2008.

[3] P. Jokela, A. Zahemszky, S. Arianfar, P. Nikander, and C. Esteve, "Lipsin: Line speed publish/subscribe inter-networking," in *Proc. ACM SIGCOMM*, Barcelona, Spain, Aug. 2009.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th ACM International Conference on emerging Networking EXperiments and Technologies (ACM CoNEXT)*, Rome, Italy, Dec. 2009.

[5] N. Paskin, "Digital object identifier (DOI) system," in *Encyclopedia of Library and Information Sciences*, 3rd ed. Taylor & Francis, 2010.

[6] R. L. Rivest and B. Lampson, "SDSI - A Simple Distributed Security Infrastructure," MIT, Tech. Rep., 1996.

[7] D. Mazières and M. F. Kaashoek, "Escaping the evils of centralized control with self-certifying pathnames," in *Proc. 8th ACM SIGOPS European Workshop*, Sintra, Portugal, Sep. 1998.

[8] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "SPKI Certificate Theory," RFC 2693, IETF, Sep. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2693.txt

[9] M. Blaze, J. Feigenbaum, and A. D. Keromytis, "Keynote: Trust management for public-key infrastructures," in *Proc. Security Protocols International Workshop, LNCS*, vol. 1550. Cambridge, England: Springer, Apr. 1998, pp. 59–63.

[10] S. Sun, L. Lannom, and B. Boesch, "Handle System Overview," RFC 3650, IETF, Nov. 2003.

[11] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture," RFC 4423, IETF, May 2006.

[12] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *Proc. ACM SIGCOMM*, US, Aug. 2002.

[13] B. Ahlgren, J. Arkko, L. Eggert, and J. Rajahalme, "A node identity internetworking architecture," in *Proc. 9th IEEE Global Internet Symposium*, Spain, Apr. 2006, in conjunction with IEEE INFOCOM.

[14] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish, "A layered naming architecture for the Internet," in *Proc. ACM SIGCOMM*, Oregon, USA, Aug.-Sep. 2004.

[15] B. Ohlman, B. Ahlgren, M. Brunner, M. D'Ambrosio, C. Dannewitz, A. Eriksson, B. Grönvall, D. Horne, M. Marchisio, I. Marsh, S. Nechifor, K. Pentikousis, S. Randriamasy, R. Rembarz, É. Renault, O. Strandberg, P. Talaba, J. Ubillos, V. Vercellone, and D. Zeghlache, "D-6.1 First NetInf Architecture Description," Deliverable, 4WARD, 7th FP EU-funded project, April 2009. [Online]. Available: http://www.4ward-project.eu/index.php?s=file_download&id=39

[16] C. Dannewitz and T. Biermann, "Prototyping a network of information," in *Demonstrations of the 34th IEEE Conference on Local Computer Networks (LCN)*, Zurich, Switzerland, Oct. 2009.

[17] C. Dannewitz, "Augmented Internet: An information-centric approach for real-world / Internet integration," in *Proc. Int. Workshop on the Network of the Future*, Dresden, Germany, Jun. 2009, in conjunction with IEEE ICC.