SoK: Public Key and Namespace Management in NDN

Pouyan Fotouhi Tehrani
pft@acm.org
Weizenbaum Institute / Fraunhofer FOKUS
Berlin, Germany

Eric Osterweil
George Mason University
Fairfax, VA, USA
eoster@gmu.edu

Thomas C. Schmidt
HAW Hamburg
Hamburg, Germany
t.schmidt@haw-hamburg.de

Matthias Wählisch
Freie Universität Berlin
Berlin, Germany
m.waehlisch@fu-berlin.de

ABSTRACT

Named data networking (NDN) enables scenarios where decentralized content distribution based on names is the centerpiece of networking. In this paper, we systematize two requirements to enable trust on a global scale in NDN, namespace management and public key management. We provide a framework to systematically assess and evaluate namespace and public key management systems, and relate their features to DNSSEC and Web PKI, the most prominent and accessible implementations of both building blocks on the current Internet. Our systematization of knowledge of existing approaches in NDN highlights strengths and shortcomings to derive options for future research.

CCS CONCEPTS

• **Networks** → **Naming and addressing**; **Naming and addressing**; *Network layer protocols*; *Application layer protocols*; **Network security**.

KEYWORDS

ICN, NDN, trust

ACM Reference Format:
Pouyan Fotouhi Tehrani, Eric Osterweil, Thomas C. Schmidt, and Matthias Wählisch. 2022. SoK: Public Key and Namespace Management in NDN. In *9th ACM Conference on Information-Centric Networking (ICN '22), September 19–21, 2022, Osaka, Japan.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3517212.3558085

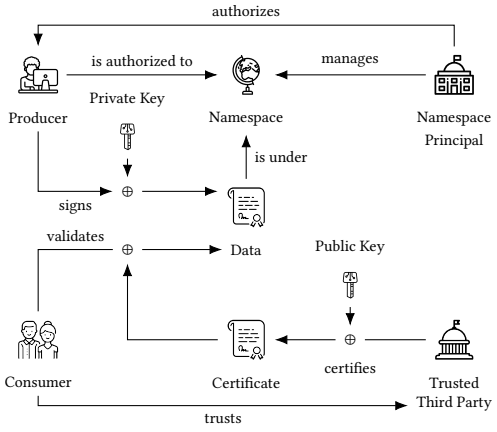

Figure 1: An overview of entities, roles, and relations in namespace and public key management in NDN

1 INTRODUCTION

Information-centric networking (ICN) is a paradigm shift in networking that centers around the idea of naming data and having the network layer handle name-based data discovery and delivery. The underlying premise of this idea is that people care about data and not its topological location in the network [35]; an idea which dates back to overlay networks such as P2P networks or the "Internet Indirection Infrastructure" (*i3*) [72]. Named-data networking (NDN) [84] is one of the most prominent and currently most lively concrete proposals of ICN.

In NDN, the name-to-data binding, as the foundation of ICN, is implemented by digital signatures over data packets that contain both the name and data in the same packet. However, as NDN names are not intrinsically bound to data, because any arbitrary producer can claim a name and bind data to it, additional effort is required to maintain *referential transparency* such that the relationship between name and data remains unambiguous. To achieve this, two further bindings are required: (*i*) name-to-producer (as name owner) and (*ii*) public-key-to-producer (as content generator). These bindings rely on two buildings blocks. A namespace management system is assumed that allocates namespaces (*e.g.,* `/org/ietf/*`) to producers and enables consumers to verify the ownership of a name, and a public key management system to bind producers (and possibly other roles such as namespace principals) to public keys, to define key life cycle management and trust models. Authenticating a name-to-data binding then succeeds transitively by first authenticating whether the producer is authorized to use that name, and whether the private key used to sign the data is valid and belongs to the producer. How these building blocks are implemented in detail and which further assumptions are made may differ between NDN applications. Considering an NDN ecosystem that allows for the

ICN '22, September 19–21, 2022, Osaka, Japan

ACM ISBN 978-1-4503-9257-0/22/09.
https://doi.org/10.1145/3517212.3558085

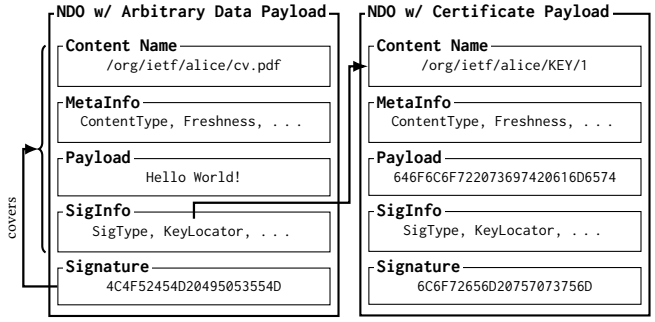

Figure 2: Examples of NDOs with arbitrary payload (left) and certificate payload (right) in NDN.

involvement of multiple stakeholders requires interoperable design choices, though.

In this paper, we investigate trust in named-data networking by systematizing knowledge about namespace and public key management in NDN. Our analysis is based on surveying more than 30 NDN applications introduced between 2015 and 2021 at common ICN publication venues. We contextualize design choices and requirements to derive a framework that may serve to compare current and future NDN applications and derive a common understanding of namespace and public key management in NDN.

The remainder of this paper is structured as follows. We start with a brief overview of NDN in Section 2. In Section 3, we discuss the relation of namespace and public key management to name-to-producer and public-key-to-producer bindings in NDN, and propose two taxonomies to better understand these two aspects and their roles in bringing trust to NDN. In Section 4, we study, assess, compare, and discuss pitfalls of existing approaches of namespace and public key management for NDN. We conclude our paper in Section 5.

2 BACKGROUND ON NAMING AND SECURITY PRINCIPLES IN NDN

The main purpose of a named data network is to find data objects based on their names in a given context. Various naming schemes, defining flat or hierarchical namespaces [16], have been proposed by different ICN implementations, each with its own set of unique features [2, 11, 22]. Names, at the core of ICN, are bound to data objects (*i.e.,* metadata and payload) to form *named data objects* (NDO) as depicted in Figure 2. Name-to-data binding is realized by producer signature over an NDO which also caters for data authenticity and integrity. The basic premise here is that a producer who publishes an NDO under a given name is authorized to use that name. A producer is identified by its signature which can be authenticated by the producer certificate. In this section, we give a brief introduction on naming and namespace in NDN as well as NDN security measures based on digital signatures (public-key cryptography).

Naming and namespace structure. NDN uses a hierarchical naming structure similar to domain names (see Section A.1 for an overview of DNS and its security extensions). An NDN name is composed of *components* [58] (comparable to labels of a domain name) that are separated by a forward slash. For example, the domain name `ietf.org` can be *ndnified* [3] as `/org/ietf`. NDN does not put any constraints on names or their global uniqueness [85], and explicitly leaves namespace management out of its specification [84]. Applications can freely define their own namespaces by deciding on structure, *e.g.,* number and order of components, and semantics, *e.g.,* using functional or constant name component of names (see Table 1). Namespaces with a limited scope, *e.g.,* local scope of smart home applications, can be managed by a single entity, whereas global namespaces, *e.g.,* used in routing and forwarding algorithms, need coordination among various stakeholders to avoid naming conflicts. Retrieving data on a global scale requires globally unique names [85].

In contrast to flat namespaces [20, 22, 39], the binding between data and name of the data is neither self-authenticating nor self-certifying in NDN. NDN needs extra measures to authenticate data objects.

Object security to verify data integrity. Digital signatures in NDN build the basis for object security. Object security allows for securing data regardless of its location or transmission channel. The minimal security assurance guarantees data integrity through cryptographic hashes [48]. The security model of NDN relies on this approach by requiring all NDOs to include a signed hash [84, 89] covering an NDO name, metadata, signature information, and its payload (see Figure 2). By computing the digest of an NDO and comparing its signed hash, a consumer can verify that the NDO has not been tampered with, *i.e.,* neither the payload nor its name or metadata has been modified. Note that only a signature by an authorized producer is acceptable as a proof of data integrity.

A producer is identified by its certificate. Certificates in NDN are special data packets that carry a public key as payload and are identified by the following naming convention: `/<IdentityName>/KEY/<KeyId>/<IssuerId>/<Ver>` [56]. The `IdentityName` is comparable to subject name of X.509 certificates [19, 34] and serves to identify the certificate holder. Every signed data packet includes a `KeyLocator` field [57] that contains the name of the certificate which can be used to authenticate data signature. For certificates, the `KeyLocator` field contains the certificate name of the issuer. Note that a certificate is published under the namespace of its owner denoted by the `/<IdentityName>` prefix. This implies that the certificate issuer is authorized to generate and sign NDOs under the special prefix `/<IdentityName>/KEY` that it does not own.

Establishing trust in NDN resembles SDSI [65] by putting the burden of defining trust anchors and trust relations on each involved entity [89]. It is noteworthy that this approach fundamentally differs from the most prominent deployment of public key infrastructure (PKI) on the current Internet, the Web PKI (see Section A.2 for an overview). Beside authentication, public key cryptography is used by NDN applications for encryption and access control purposes (see Table 1).

3 PROBLEM STATEMENT: TRUST IN NAMED DATA NETWORKS

Authenticating name-to-data bindings in NDN presupposes authenticating name-to-producer and public-key-to-producer bindings. These two bindings build the basis of trust in NDN by allowing

Table 1: An overview of selected CCN/NDN applications and their namespace and key management requirements, based on surveying research published at ACM ICN '15–'21

		Namespace Requirements			Key Usage		
	Name	Prefix	Functional Components	Name Format†‡	Confidentiality	Authentication	Access Control
Routing and forwarding	NLSR [31]	Network name	Site and router names	/<network>/<site>/<router>	–	Routing messages	–
	LSCR [29]	Network name	Site, router, msg type	/<network>/<site>/<router>/**LSCR**/**LSA**/<typeID>	–	–	–
	SNAMP [5]	Global prefix	–	/<network>/<site>/	–	Link objects	–
	MNDN [55]	1. Global prefix 2. Name server	– DFZ prefix	/<network> /**GNRS**/<DFZ-prefix>	–	Link objects	Zone mappings
	KITE [88]	Global prefix	Tracing segment	/<network>/<traceSeg>	–	Trace interest	–
	LEO NDN[45]	Satellite location	–	/<baseNS>/<satID>	–	–	–
Sync	ChronoSync [90]	1. Broadcast space 2. –	Sync interest Sync reply	/<broadcast>/<appName> /<producerID>/<appName>	Sync data	–	Sync group
	PSync [86]	Multicast space[1]	Sync interest and reply		–	–	–
	MMORPG Sync [52]	Game ID	Game instance	/<appID>/<gameInst>	–	–	–
Security	ICN-based MIS[9]	Identity	Application ID	/<idPart1>/<idPart2>/<appID>	–	Identities	Data
	CCN-AC[43]	Anonymizer domain	Parameters	/<anonDomain>/[<encName>|<cmd>]	Interest/Data	Anonymizer/Caches	Interest/Data
	NDN OCSP [64]	1. Query service 2. Update service	– key ID and Update commans	/<ocspNS> /<server>/<keyID>/<cmd>	–	Services	Update service
	NDN-ABS [63]	–	ABE public params	/<baseNS>/**ABE**/<public-params>	Data Packets	Producer	Consumer
Diagnostic	NCMP [49]	–	Command and params	/<baseNS>/**register**/<cmd>	Result	Requester	Server
	NDN-Trace [38]	Trace prefix	Parameters	/**Trace**/<pathType>/<traceType>/<name> /<nonce>/<FaceID>	–	–	–
	DNMP [61]	Network root	Parameters	/<root>/dnmp/<params>	(Replies)	Commands	Commands and replies
RPC	NFaaS[42]	Execution prefix	Parameters	/**exec**/<appClass>/<func>/<input>	–	Requester	Functions
	RICE [40]	1. Function name 2. Instance name	parameters state	/<func>/<params> /<instID>/<func>/<state>	(Input)	Results	(Server)
	CFN [41]	Node name	Parameters	/<nodename>/<framework>/<params>	–	–	–
	IceFlow [44]	Application name	Dataflow parameters	/<app>/<actor>/<instance>/data/<partition>/<object>	–	–	–
IoT	Multi-Source IoT [6]	Multi-source ID	producer/device ID	/<baseNS>/<msINT>/<deviceID>	–	–	–
	Keyword-based ICN-IoT [10]	–	Function name and hashtags	/<prefix>/<func>/<hashtags>	(Data)	(Users)	(Devices)
	IoT On-boarding [17]	Network root	Commands and parameters	/<root>/<cmd>/<params>	Data Packets	Devices	–
	IoT QoS [26]	–	Traffic class	/<baseNS>/<trafficClass>	–	–	–
	FW Update IoT [25]	Deployment ID	Globally unique vendor / device class	/<deplID>/<vendorID>/<devCls>	–	Vendor identity	–
Misc	NDN-RTC[24]	–	Steam and packet params	/<baseND>/**streams**/<streamID>/<threads>/**frames**/<packetType>/<frame> /<dataType>	–	Producer	–
	CNS [15]	Organization	1. Administrative ID 2. Incident reponse ID	/<org>/**authorities**/<adminID> /<org>/**incidents**/<incidentID>	–	–	–
	TCP/ICN [51]	1. Forw. Proxy prefix 2. Rev. proxy prefix	TCP payload TCP headers	/<prefix>/<tpc4tuple>/<seqNo>/<wrapNo> /<prefix>/<tcpHeaders>/<nonce>	–	–	–
	NDN De-Lorean [81]	–	Chronicle Tree	/<service>/<type>/<state>/<index><digest>	–	Service timestamps	–
	NDNFit [82, 83]	Trust root	User, device, and app IDs	/<root>/<user>/<dev>/<app>	Content	Producer	Userspace
	HPNM [77]	–	Video encoding parameters	/<baseNS>/**SVC**/<name>/<params>	–	–	–

† For the sake of simplicity name formats only represent most significant prefixes of names and omit the rest (*e.g.,* segment component)

‡ Notation: <> denotes a functional component that can be composed of multiple specific name components, [] denotes alternative components (separated by |), and constant components are in **bold** type.

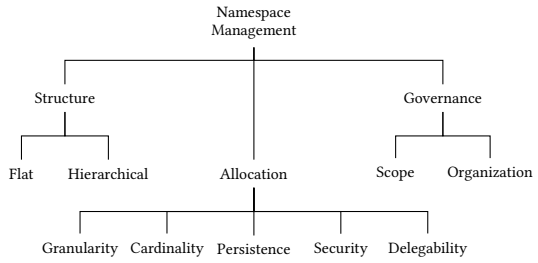

Figure 3: A taxonomy of namespace management

a data consumer to verify that a name is not spoofed, and an authorized producer has generated the content bound to that name. Achieving a common understanding of trust in NDN requires consensus among producers, consumers, and other involved nodes on the network regarding the context that defines these bindings.

3.1 Name-to-producer Binding and Namespace Management

Binding a name to a content producer requires authorization of a producer to publish data under that name. In contrast to self-certifying or self-authenticating names, names are arbitrary in NDN and an external mechanism is needed to verify the ownership of names. This includes namespace management to allocate names to their legitimate owners (see Bechtold [12]) and functions that enable consumers to securely validate name assignments. In Figure 3, we propose a concise taxonomy of namespace management adapted from prior work [12, 16, 37, 66, 70] based on *structure*, *allocation*, and *governance*. After defining each aspect, we also briefly discuss the pitfalls in each category that need to be considered when designing namespace management for NDN.

Structure. A namespace structure can be *flat* or *hierarchical*. DNS, for example, uses a hierarchical naming structure which is divided into sub-namespaces or zones. Each zone can define its own set of structuring rules. For example, the `.us` top-level domain (TLD) defines a rigorous structure up to fourth level in its hierarchy [18]. NDN uses hierarchical names not only to improve interest forwarding and data delivery (*e.g.,* through prefix aggregation) but also to cater for semantically meaningful names, *e.g.,* as the basis for its security approach (see Section 2). In addition to its syntactical structure, a namespace can be structured along semantic meanings of name components. For example, ChronoSync [90], a synchronization protocol for NDN, defines a broadcast namespace (*e.g.,* `/ndn/broadcast/chronos/chat`) to synchronize the state of messages published by producers under their own namespaces (*e.g.,* `/wonderland/alice/chronos/chat`).

Many NDN applications drive the namespace structure to ease the development of applications (*e.g.,* see Thompson, Gusev, and Burke [75]). Using functional name components [60] to convey specific semantics leads to larger implication in global settings, though, in particular when functional components are shared among multiple producers. In our previous example, ChronoSync, having a broadcast namespace requires different producers to be authorized both to a broadcast namespace and their own namespaces at the same time. If names-to-producer binding succeeds through certification of a producer public key (as we discuss in the next section), either all producers must share the same key to announce their local state changes under the broadcast namespace, or have their public key certified both by the broadcast namespace owner and the namespace principal of their own namespaces.

Allocation. Allocation describes how names are allocated to namespace principals in terms of *granularity* (*i.e.,* allocation unit), *cardinality* of namespace to principal allocations, validity period (*persistence*), *security* of allocation method, and possiblity of delegating a namespace to a third party (*delegability*). In DNS, for example, a complete sub-namespace, a zone, is delegated to exactly one zone owner, who can further delegate any part of its own namespace. DNS allocations are not persistent, and zone ownership can vary during time. The security extensions of DNS (see Section A.1) are used to cryptographically secure delegations. On the top level of DNS hierarchy, *e.g.,* `.gov`, allocations might be subject to specific eligibility requirements and rules defined by the namespace governance. Similar to zone delegations in DNS, In NDN a complete namespace denoted by a prefix is allocated instead of single names.

Governance. We consider the set of rules, policies, and politics around names and naming as namespace governance. To illustrate this, we refer to the history of DNS and its evolution from local `HOSTS.TXT` files to a globally distributed database governed by the *Internet Corporation for Assigned Names and Numbers* (ICANN). During this process the management of domain names, which mainly used to be of technical character, was enhanced with policies agreed upon among various stakeholders, and national and international laws to regulate naming on the Internet, *e.g.,* to address naming conflicts due to trademark violations. Names in DNS, or more specifically namespaces in terms of zones, are allocated from ICANN to registries to a single logical principal, the zone owner. Khare [37] argues that such one-to-one mappings create a degree of scarcity with political consequences, observed during the DNS reform and emergence of ICANN. The importance of ICANN and more specifically its role with respect to transparency and accountability in establishing trust on the Internet is also acknowledged in a report by the Global Commission on Internet Governance [23]. Beyond DNS, Bechtold [12] argues that designing and controlling a namespace in general brings about "politics, policy, and regulation". Due to its similarity both in terms of structure as well as semantics to the DNS, we argue that an NDN namespace also requires proper governance to enable Internet-scale scope (see our previous work [74] for a thorough discussion).

Occasionally, it is required to have a set of names, addresses, *etc.* confined to a restricted context, *e.g.,* a local or private network. The NDN forwarding daemon (NFD), for example, uses the `/localhost/nfd` namespace for message exchange as part of its management protocol [59]. Such namespaces must also be defined as part of a namespace management governance.

3.2 Public-key-to-producer Binding and Public Key Management

Digital signatures build the basis of the NDN security model. As producers are bound to public keys through certificates, securing NDOs presuppose public key management. We propose a taxonomy for

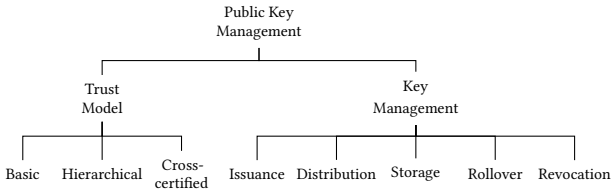

Figure 4: A Taxonomy for Public Key Management

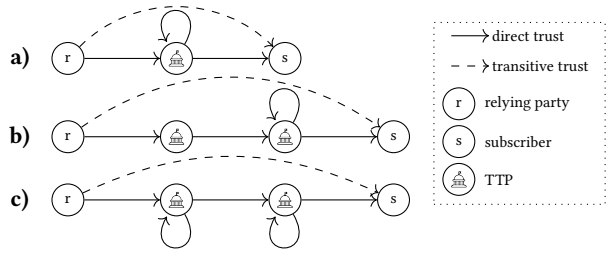

Figure 5: Public key management trust models—a) basic, b) hierarchical, and c) cross certified (loops denote trust anchors)

public key management based on *trust model* and *key management* (see Figure 4).

Trust Model. One of the main challenges in public key cryptography is the establishment of trust in the public key. Various trust models have been proposed to avoid manual out-of-band key exchange and verification by introducing trusted third parties (TTP) that generate and store key pair (*e.g.,* key escrow systems) or certify authenticity of a public key (*e.g.,* certification authorities). Here, a relying party (RP), *i.e.,* an entity that relies on the information included in a certificate to make a decision, for example, a data consumer, only needs to trust a limited number of TTPs and assess the trustworthiness of any public key based on policies of this trust model. Figure 5 depicts various trust models adapted from [36].

To establish TTPs, a relying party first needs to retrieve a set of trusted keys, denoting trust anchors (TA), in a bootstrapping phase. A trust model can define how trust is then deferred from a TA to any other TTP [13]. In DNSSEC PKI [8], for example, a single TA (root zone) is defined but RPs can decide to manually trust any other zone as their TA, whereas in Web PKI any CA that fulfills a set of requirements can be accepted as TA by operating system or browser vendors as maintainers of a *trust store* [21] (see for example Mozilla CCADB [54] or Apple root Certificate Program [7]). In contrast to DNSSEC, CAs in Web PKI are not limited in their certification so that a single compromised CA can issue certificates for any arbitrary domain name and any identity. To counter this, Certificate Transparency (CT) logs [68] were introduced so that anyone can track which CAs issues certificates for which domain names and subjects. Similar approaches to CT logs have been also proposed for NDN [14]. However, as we see later in Section 4, public key management solutions for NDN which unify the role of namespace principal and CA for a given namespace, are less dependent on such certificate logs. This is due to the fact that in such cases the relationship between a certification entity, *i.e.,* namespace principal, and the namespace it is authorized to certify is unique and unambiguous. So if an entity other than the namespace owner issues certificates for that namespaces, those certificates are by definition invalid and that entity is evidently misbehaving. Certificate logs here can, nonetheless, be used by namespace principals to detect misbehaving parent principals, *e.g.,* when a namespace principal delegates the same sub-namespace to two different owners.

A trust model also defines how valid trust chains between trust anchors and any arbitrary key can be constructed. In DNSSEC, there exists only a single verifiable path from any given key to a TA which is constructed from that TA, *e.g.,* root, to respective zone along the delegation path. In contrast, the cross-certification model of Web PKI with multiple independent trust anchors allows multiple valid trust chains that cannot be reconstructed by an RP without additional information from the subscriber. To address this ambiguity, a web server provides the complete trust chain to its client. Trust chains in NDN are built from bottom to top through iterative following of `KeyLocators` and as such pose a surface for availability attacks in which an adversary can trick an RP to start building a trust chain that does not terminate with a TA or contains loops, as we discuss in more detail in Section 4.

Key Management. Procedures and policies to create, distribute, store, rollover, and revoke keys are referred to as key management [46]. Most commonly, keys are generated by their owners and are only certified by a trusted entity. Key escrow systems in which a TTP is also responsible to generate private keys are an exception to this rule. In such systems the TTP generates both private and public keys practically defeating the non-repudiation property of digital signatures and creating a single point of failure in case of compromise.

Key distribution defines how RPs can retrieve public keys. In DNSSEC, keys of a DNS zone are bundled together as a `DNSKEY` RRset, in Web PKI, the web server provides its certificate (signed public key) to clients, CAs provide repositories that contain certificates of intermediate and root CAs, and browsers or operating systems maintain *trust stores* that contain all trusted CAs. In NDN, a certificate needs to be retrieved by its name just like any other NDO. The `KeyLocator` field of an NDO defines the name of certificate that can be used to authenticate that NDO.

Key rollover refers to the procedure of transitioning an old key with a new one while maintaining the existing trust relations. In Web PKI, rollover for leaf certificates are instantaneous while trust anchors transitions succeed out-of-band by trust store maintainers despite procedures defined for Root CA key rollovers in RFC 4210 [1, §4.4.]. Key rollovers in DNSSEC are defined for both TAs [71] and other keys [53].

Finally, in case of key compromises, the issuer or the key owner revokes the key to avoid its further use. Although keys can be cached and used as long as they are not expired, a relying party still needs to regularly check if a key is not revoked. Revoking keys in DNSSEC can succeed instantaneously but might lead to inconsistencies due to cached keys, specially for KSKs for which the respective `DS` record must also be updated in the parent zone (see Osterweil [62] for a thorough discussion). Wang and Xiao [76], for example, propose a method to roll over KSKs in emergency situations without introducing temporal inconsistencies. In Web

PKI, the revocation status of a certificate can be verified through Certificate Revocation Lists (CRL) [19] or the Online Certificate Status Protocol (OCSP) [67]. Analogous solutions for NDN has also been proposed such as suicide directories [79], distributed OCSP for NDN [64], or the ledger-based revocation approach *CertRevoke* [78].

Improper key rollover and revocation can have negative security implications. RFC 7583 [53], for example, discusses how to roll over DNSSEC keys without introducing inconsistencies in the network considering possibly cached records and propagation delays. In NDN certificate validity is temporally bound. And the validity of a certificate is bound by the validity of other keys in its trust chain. The validity of a data packet, consequently, is determined implicitly by the validity of its producer certificate. Such temporal dependency not only impacts producers and consumers but also intermediate and caching nodes. Rolling over or revoking a key equals invalidating all data signed by that key. If a trust anchor or intermediate CA rolls or revoke their keys before they are expired, an adversary can use old, yet valid, keys for replay attacks and disturb availability. Even data and certificates that are cached on the network can lead to failed authentication in case of ill-timed revocation and rollovers. The validity period of NDOs in general also impacts caching. In NDN, caches keep an NDO for at least until its *freshness period* (a relative value) expires. It is, however, reasonable to purge an NDO from caches as soon as it expires even if it still is considered to be fresh.

4 NAMESPACE AND KEY MANAGEMENT APPROACHES FOR NDN

In this section, we review different namespace and public key management schemes proposed for NDN, describing each according to the taxonomies introduced in Section 3 and summarized in Figures 3 and 4.

4.1 NDN Public Key Management (PKM)

Overview. In an NDN technical report, Yu [79] proposes a public key management method for NDN which also allows integrating namespace management based on naming semantics of certificate issuance and ownership. The author defines the NDN certificate format (preceding the current format [56]) and procedures to issue, verify, and revoke signatures and keys. Here, a certificate name is formatted as `/<identity>/KEY/<keyID>/<ver>`. The `identity` component serves to identify the public key owner, *e.g.,* `/org/ietf/alice`. The process of certification implicitly delegates the `/<identity>` namespace to the certificate owner to publish its own data, *e.g.,* revocation packets, as discussed below. To allow for multiple keys to be bound with the same identity, each certificate name is carries a `keyID` which (similar to a `DNSKEY` key tag) can be used to distinguish keys of the same owner. A version number is also included in a certificate's name in case the same public key is re-signed. The identity prefix is divided from key id and version number with a constant `KEY` name component. This approach has been (partly) implemented by NDN applications, *e.g.,* NDNfit [82].

Challenges. Similar to the current naming convention for NDN certificates, this approach requires that the issuer is authorized to publish under the namespace that actually belongs to the key owner, *e.g.,* under `/org/ietf/alice/KEY` prefix. Two possible alternatives can be imagined: (*i*) the issuer is both the namespace principal and the certification authority of the parenting namespace, *e.g.,* `/org/ietf`, or (*ii*) identities are prefixed with a namespace delegated to the issuer, *e.g.,* `/letsencrypt/org/ietf/alice`. Whereas the latter approach requires a separate infrastructure for namespace management, the former solution (as used in NDNFit [82]) entangles namespace management and public key management, unifying the role of namespace principal and certification authority. Here, namespace delegation succeeds through certification, *i.e.,* the owner of `/org/ietf` namespace can delegate `/org/ietf/alice` to Alice by signing her certificate under `/org/ietf/alice/KEY` namespace (see Figure 2). Alice, as her own namespace principal and CA, can in turn delegate any name under her namespace. Note that although certification and key management are defined along the hierarchical naming structure of NDN, the author does not further specify name and certification delegations from the root, *i.e.,* the '/' namespace.

The pitfall of unifying the role of namespace principal and CA is that any data producer who owns its dedicated namespace must also take care of key management. To take the burden of certificate distribution from the certificate owner, the author foresees *certificate hosts* which act as always-on, long-time caches for certificates. An owner can then designate certificate hosts to distribute its certificate(s) and is also responsible to make sure that the latest certificates are always stored by the certificate hosts. Yet, key life cycle management remains a duty of its owner. To revoke a certificate, its owner publishes a revocation certificate, *i.e.,* a self-signed version of the certificate, under the same name but with an appended `REVOKED` name component. Certificate owner must then actively respond to interests for the revocation status, *e.g.,* interests for `/org/ietf/alice/KEY/1/1/REVOKED`, using a signed NACK if the certificate is not revoked. The fact that the key owner and not the issuer is responsible for revocation and issuing signed NACKs can be advantageous for an adversary. Consider that Eve has gained access to Alice's private key. Alice now wants to revoke its key but Eve can suppress revocation certificates if she manages to publish signed NACKs before interests for revocation status reach Alice. An alternative approach proposed by the author is to have a global "suicide directory" to register all revoked certificates resembling a combination of Certification Revocation Lists (CRL) [19] and Certificate Transparency Logs [68] of the Web PKI.

Beside revoking keys, single signatures can also be revoked in this approach. Given an NDO, the signer needs to regularly publish "signature status" under `/<identity>/SigStatus/<hash>/<ts>` with `hash` being the digest of signed NDO and `ts` the publication timestamp of the status. The content of the status denotes if the signature is revoked. Producers can delegate this process to a third party through a special certificate extension which is not further detailed by the author. The downside of this is that publishing any NDO obligates its producer to regularly publish a respective signature status *even* if no consumer is currently requesting that NDO. An alternative would be to extend the `SigInfo` of an NDO (see Figure 2) to contain a validity period similar to the validity period of DNSSEC signatures (`RRSIG`; see Section A.1). Although this alternative could increase the network traffic, it would bring two advantages: (*i*) the producer needs to sign or re-sign data only if it

Table 2: Simplified trust schemata and exemplary names for `/org/ietf` namespace with a single trust anchor

Rules

#	Name pattern (data or certificate)		Name pattern (certificate)
1	`<org><ietf><KEY>[id]`	→	`/org/ietf/KEY/1`
2.1	`<org><ietf><admin>[user]<KEY>[id]`	→	`<org><ietf><KEY>[id]`
2.2	`<org><ietf><admin>[user]<KEY>[id]`	→	`<org><ietf><admin>[user]<KEY>[id]`
3	`<org><ietf>[user]<KEY>`	→	`<org><ietf><admin><KEY>[id]`
4	`<org><ietf><~>([user])<>*`	→	`<org><ietf>([user])<KEY>[id]`
5	`<org><ietf><public><>`	→	`<org><ietf>[user]<KEY>[id]`

Instances

#	NDO Name		Key Locator
1	`/org/ietf/KEY/1`	→	`/org/ietf/KEY/1`
2	`/org/ietf/admin/bob/KEY/1`	→	`/org/ietf/KEY/1`
3	`/org/ietf/admin/eve/KEY/1`	→	`/org/ietf/admin/bob/KEY/1`
4	`/org/ietf/alice/KEY/1`	→	`/org/ietf/admin/bob/KEY/1`
5	`/org/ietf/~/alice/cv.pdf`	→	`/org/ietf/alice/KEY/1`
6	`/org/ietf/public/party.pdf`	→	`/org/ietf/alice/KEY/1`

is actively consumed, and (*ii*) the validity period of an NDO would define a hard and absolute deadline for caches to remove the NDO.

Finally, given an NDO, the process of validation succeeds as follows:

(1) Verify that NDO name shares the same prefix with the certificate of the signing party as denoted in `KeyLocator` (assuming that certification and namespace delegation are unified)
(2) Fetch certificate and verify that it has not expired, *i.e.,* the `ValidityPeriod` holds
(3) Verify that the certificate is not revoked either by:
  (a) Consulting the suicide directory or
  (b) Confirming that no revocation certificate exists (signed NACK with an un-revoked signature; see above)
(4) Verify that the NDO signature is not revoked
(5) Validate the NDO signature using fetched certificate

This procedure is then repeated for any intermediate certificate (an NDO itself) by following the certificate chain until a trust anchor is reached. The validation starts with an NDO, moves along the delegation path, and ends with a trust anchor. To avoid multiple round trips to authenticate a single packet, the author introduces *key bundle* which contains all certificates required to validate an NDO. A key bundle, however, does not obviate the need to verify revocation status of included signatures.

4.2 Trust Schema

Overview. Based on the observation that different applications might need to define different trust relationships, Yu *et al.* [80] propose "trust schemata", a name-based trust management system for NDN. The idea of trust management can be traced back to Blaze, Feigenbaum, and Lacy [13], who considered existing trust mechanisms, *e.g.,* PGP, as too narrow and inflexible and introduced a language to express security policies, to validate if a credential fulfills a policy, and to "defer trust" to third parties. As an established part of NDN security specification, schematized trust in NDN is a trust management system that defines both namespace as well as public key management in terms of trust rules and trust anchors. A trust rule is a mapping that defines the relation between an NDO name and the certificate name of its signing party. Namespace authorization in this approach are defined by rules that map non-certificate name patterns to certificate name patterns, and public key management by rules that map certificate name patterns to another certificate name pattern. Trust rules can be linked together to form a trust chain ending with a trust anchor. A trust anchor is denoted when a name pattern is mapped to a (set of) specific certificate name(s). A trust anchor here denotes both a namespace principal and a CA.

An example of trust schema is provided in Table 2. In this simple scheme, Rule #1 defines a single trust anchor that controls `/org/ietf` namespace. Rules #2 and #3 define administrators as CAs for the producers authorized to names under *user* (`/org/ietf/~/[user]`) and *public* (`/org/ietf/public`) namespaces. And finally, Rule #4 and #5 authorize producers to publish under these namespaces.

Advantages. The main advantage of trust schemata is their expressiveness in defining namespace and key management policies. For instance, in our previous example, an administrator can only certify public keys and is not authorized to any namespace to publish non-certificate NDOs, while a non-administrator user may publish NDOs under specific namespaces but cannot delegate sub-namespaces or certify other users. Furthermore, as namespace and key management are not coupled, a single entity (using a single certificate) can be authorized to multiple namespaces (Rules #4 and #5).

Challenge: Key Revocation. Decoupling namespace and key management require careful design of trust rules to cater for certificate (or signature) revocations while avoiding potential security issues. In the following, using the example of revocation methods proposed by NDN PKM (from the same author, see Section 4.1), and CertRevoke [78], we discuss challenges of catering to key revocation when designing trust schema.

We start with a simple trust schema as depicted in Table 2. When using NDN PKM, we see that a certificate owner is not allowed here to publish a revocation notification, thus, leaving the burden of revocation to the issuing CA. The pitfall, however, is that a (compromised) CA can revoke a certificate that it has not issued and still fulfill the trust rules. For self-referencing certification rules (*e.g.,* Rule #2.2), this property allows revocation of peering CAs as well. To elaborate this we use the example of instances in Table 2. It can be seen that Alice cannot revoke her certificate by publishing a self-signed certificate under `/org/ietf/alice/KEY/1/REVOKED` as she is not authorized to publish under that namespace. She also cannot publish signed NACKs to indicate that her certificate is still valid. At the same time, Eve can publish a revocation certificate for Alice (Instance #3) despite that fact that Bob has issued the certificate (Instance #4). Eve can also even revoke Bob's certificate (Rule #2.2.) or suppress revocation certificates with signed NACKs in response to revocation status requests. Adding an extra trust rule to allow for self-signing revocation notifications as follows:

`(<>*)<KEY>([id])([ver])<REVOKED> → (<>*)<KEY>([id])([ver])`

would solve these issues without giving any unnecessary privileges to a certificate owner. Note that such rule still does not solve potential availability attacks discussed in Section 4.1. Another solution is proposed by CertRevoke is to explicitly include the revoker identity in a revocation packet and then utilize fine granular trust schema rules to limit certificate revocations to their exact issuers or owners. For example, the revocation message for a certificate under `/org/ietf/alice/KEY/1/issuer1` (see Section 2) can only carry the name `/org/ietf/alice/REVOKE/1/[issuer1|self]` and be signed by the issuer (if last component is the `issuer1`) or the owner (denoted by `self` name component). The downside of CertRevoke is that to check the revocation status, a relying party needs send out two messages (one with issuer ID and one with `self`) to make sure that neither has revoked the certificate.

Challenge: Authentication. Assuming that revocation procedures are defined, authenticating an NDO using trust schemata is similar to the procedure discussed previously (see Section 4.1) but instead of step 1, a relying party need to verify that the name of an NDO and its `KeyLocator` match at least one trust rule.

Furthermore, no a priori known authentication path may lead to unstable behavior. As the trust chain is built from leaf to root (in contrast to DNSSEC) but the authentication the other way around, an adversary can deceive a relying party to construct a trust chain that conforms to trust rules but do not end with a trust anchor. Self-referenced trust rules even allow creating loops and increase the length of the trust chain. This poses a serious attack surface for availability. For example, an adversary can publish an NDO under Alice's namespace signed with a fake certificate `/org/ietf/alice/KEY/2` attributed to Alice. This certificate can then be signed by an administrator whose certificate is signed by another fake administrator and so on. A relying party would then start authenticating the fake NDO by constructing a trust chain that never terminates. This problem can be solved by bundling the trust chain with an NDO (see key bundle in Section 4.1).

Challenge: Scoping and Synchronization. Finally, Trust schema suffer from two additional major shortcomings. First, the limited scope. It is not possible to delegate trust management, *i.e.,* having another entity define trust schema for a sub-namespace. Consequently, a relying party need to fetch the set of trust anchors and trust rules for every namespace that it aims to validate NDOs from. Second, no possibility of synchronization. If the trust schema of a namespace is changed at some point in time, there is no way for a relying party to know which set of rules apply to which set of NDOs. There is no temporal binding between an NDO and respective trust schema.

4.3 NDNSSEC

Overview. Using the example of DNS and observing changes in the past decades, Tehrani *et al.* [74] argue that namespace management, specially on an Internet scale, requires attending to organizational and political aspects. Based on this, the authors suggest using the DNS namespace for NDN to take advantage of existing infrastructure without a need to establish organizations, policies, *etc.* to attend non-technical aspects (*e.g.,* an equivalent of ICANN). Additionally, the proposed solution uses the DNSSEC PKI to secure namespace bindings and is also used for key management in NDN. Acquiring a namespace in NDNSSEC then becomes equal to a DNS zone delegation. For example, the zone owner of `www.ietf.org` is also the namespace principal of `/org/ietf/www` on NDN. Note that, although approaches to distribute keys and other data using structures similar to DNS has already been proposed before (see for example NDNS [3] and KRS [47]), NDNSSEC emphasizes on taking advantages of established technical and non-technical aspects of DNS and DNSSEC [74].

To realize key management, the role of namespace principal and CA for a namespace are unified. To authorize producers to publish NDOs, a namespace principal includes respective keys in its zone's `DNSKEY` RRset (see Section A.1). In this approach, the name of an NDO and its `KeyLocator` share the same prefix that is the NDN equivalent of fully qualified domain name of the zone apex. The `KeyLocator` field uses a simplified NDN certificate naming (see Section 2) with `KeyId` being the digest of a `DNSKEY` and empty values for `IssuerId` and `Ver` name components. To fetch keys over DNS, a consumer can either run a local DNS resolver [73] or rely on a designated trusted third party (TTP) for key retrieval and its translation into NDN certificates. The latter solution allows having namespaces which are only valid within a private network [74], but also allows the local resolver to publish certificates under namespaces that it does not technically own. Regardless of how DNS records are introduced to NDN, *i.e.,* through a bridge resolver or an authentication chain, the lifetime of a key is defined by its covering signature, *i.e.,* `RRSIG` (see Section A.1). Respectively the certificates that are generated by the bridge resolver must reflect the validity period of the `RRSIG` that covers the containing key.

Challenge: Maintaining Trust Chains . To avoid retrieving DNS records over the Internet, the authors suggest that zone owners publish the complete authentication chain for their zones (similar to key bundles in Section 4.1) or to mirror DNS data using NDNS [4]. The former option, however, requires zone owners to publish a new chain every time any of the included records in the chain is invalidated (recall that DNSSEC signatures have a validity period) or a key is revoked. Furthermore, a zone owner must regularly scan for changes in its parenting zone as DNS(SEC) does not define a method to signal such changes. Authentication chains, just like certificates, must also be enhanced with a validity period which expires no later than the earliest expiration data of all involved signatures.

Key revocation in NDNSSEC is defined as removing a key from the DNS zone. And signature revocation for certificates is not required for keys, as each DNSSEC signatures (`RRSIG`) already carry a validity period that semantically corresponds to a "signature status" (see Section 4.1). Another advantage of DNSSEC is the set of well-defined key rollover procedures [53, 71] that also define rollover procedures for trust anchors that spares the need for out-of-band key exchange after an initial trust bootstrapping.

Challenge: Authentication and Authorization. Given an NDO, the authentication process is similar to the previously discussed procedure (see Section 4.1), but instead of step 2 and step 3, either a bridging reolver fetches and validates a certificate or an authentication chain containing the key is retrieved and validated. In the latter approach, the `KeyLocator` field carries the name of the authentication chain instead of a certificate.

The major drawback of NDNSSEC is caused by the fact that not single records, *e.g.,* a single `DNSKEY` record, but the set of same-type resource records are signed together. To validate if a producer is authorized to publish under a namespace, thus, the keys for all authorized producers must be retrieved to validate their covering signature. This poses a major scalability issue as the number producers for a namespace increases. To authorize a single producer to multiple namespaces, its public key must be replicated to all respective zones. Another shortcoming of NDNSSEC is its dependence on DNS and a lack of specific solution on how to mirror DNS data in NDN ecosystem.

4.4 Identity-based Trust

Overview. Zhang *et al.* [87] introduce a method to use NDN names, or more specifically their string representation, to generate cryptographic key pairs using *identity-based cryptography* (IBC) [69]. Here, a 'private key generator' (PKG) is in charge of issuing private keys for identities. In this approach, the producer must first consult a 'Name Registration Service' (NRS) to register a desired name. If the registration succeeds and the producer is authorized, the NRS also retrieves the corresponding private key (generated using producer's identity) from the PKG and forwards it to the producer. Namespace management in this approach is centralized and handled by the NRS which also coordinates private key issuance with the PKG. Relying parties can then use the public parameters of PKG to generate public keys for identities. An identity is included as metadata in an NDO, alongside the NDO name containing the PKG parameters, and can also be part of an NDO name. However, if the identity is not explicitly part of the name, *i.e.,* no unique name-to-producer binding is given, an authorized producer can publish NDOs with valid signatures under any arbitrary name.

Challenge: Issuance and storage. Key escrow, *i.e.,* the circumstance that an entity other than the key owner generates and stores keys, is a common weakness of all IBC approaches (see Section 3). Among the proposed solution, certificateless signature scheme has recently also been proposed for NDN [32].

Challenge: Scalability. Having only one instance of NRS and PKG, is an obstacle to scalability in this approach. To address this issue, the authors propose a hybrid scheme where local instances of NRS and PKG are responsible for smaller sections of the global namespace. The public parameters of each sub-namespace is signed by a set of CAs as part of a PKI that is trusted by relying parties, *e.g.,* DNSSEC. An alternative to address scalability is proposed by Hamdane *et al.* [28] based on hierarchical IBC (HIBC). Here, instead of a single global PKG or a set of domain-local PKGs, a root PKG is designated, and key issuance can be delegated to subordinate PKGs. A public key is then calculated from a given identity combined with the identity of its ancestors.

Challenge: Key Revocation. Key revocation in IBC is its main weakness. As a unique string is always mapped to a unique public key (given a set of public parameters), and keys, public or private, are not temporally bound, a key is considered valid as long as an identity is considered valid and vice versa. Revoking a key, thus, means revoking an identity and in turn revoking all data produced and signed by that identity. To address this shortcoming, Zhang *et al.* [87] propose to append a timestamp to identities and to consider that identity as expired as soon as a predefined amount of time spans. For relying parties, however, there is no reliable way to verify if a key is revoked at any point in time (*e.g.,* compared to revocation lists), while a compromised key can sign data as long as the validity window is not expired. In case of HIBC, Hamdane *et al.* [28] note that the revocation problem can be solved by simply revoking the public parameters of the PKG which generated the compromised key. The procedure of revoking a single PKG is not elaborated by the authors, nonetheless, revoking a PKG equals to revoking all keys generated by that PKG.

Authenticating an NDO in the approach proposed by Zhang *et al.* [87] succeeds as follows:

(1) Extract the name of NDO containing public parameters from NDO's metadata
(2) Retrieve public parameters and verify its validity (signed by trusted PKG)
(3) Extract identity name from NDO's metadata and construct corresponding public key using retrieved parameters
(4) Validate NDO using generated public key

In the hybrid scheme, instead of step (2), the public parameters are fetched from a trusted pre-defined PKI. Note that this procedure does not consider signature revocations. In the hierarchical approach [28], the `KeyLocator` field contains the name of the NDO with the public parameters of responsible PKG. These parameters are signed by the parent PKG and having the root parameters suffices to generate public key for an identity on any arbitrary level of the hierarchy. To authenticate an NDO, the following method applies:

(1) Extract the name of certificate containing public parameters from NDO's `KeyLocator`
(2) Fetch certificate and verify that it has not been expired
(3) Verify that the certificate and the PKG parameters are not revoked
(4) Validate NDO signature using fetched certificate

Similar to the procedure elaborated in Section 4.1, this procedure is repeated for all intermediate certificates until the root PKG is reached. Note that signature revocation is neither addressed here.

4.5 Summary

The landscape of namespace and public key mangement in NDN is diverse and far from being standardized. We summarize our findings based on our taxonomies (see Section 3) in Table 3 and Table 4 and now discuss comprehensively pitfalls of namespace and public key management in NDN.

Namespace Management. All approaches propose a local namespace, *i.e.,* a context with limited scope, except NDNSSEC. NDNSEC relies on the DNS as its underlying namespace with globally unique names, all other solutions are only suitable for local namespace management due to lack of well-defined name allocation and collision avoidance on a global scale. All but IBC and HIBC cater for allocating names to multiple producers. Identity based namespace management alongside trust schemata define permanent name allocations which cannot be delegated further. To void allocations in (H)IBC, either the PKG must rollover its parameters (both public and private) and consequently void all its generated keys (as proof

Table 3: Summary of namespace management approaches in NDN based on taxonomy in Figure 3

		Allocation					Governance	
	Structure	Granularity	Cardinality	Persistence	Security	Delegability	Scope	Organization
NDN PKM [79]	Hierarchical	Subspace	1-n	Transient	TA Signature	✓	Local	Centralized
Trust Schema [80]	Hierarchical	Arbitrary	1-n	Permanent	TA Signature	✘	Local	Centralized
NDNSSEC [74]	Hierarchical	Subspace	1-n	Transient	DNSSEC PKI	✓	Global	Centralized
IBC [87] & HIBC [28]	Hierarchical	Subspace	1-1	Permanent	PKG Signature	✘	Local	Centralized

Table 4: Summary of public key management approaches in NDN based on taxonomy in Figure 4

		Key Management				
	Trust Model	Issuance	Distribution	Storage	Rollover	Revocation
NDN PKM [79]	Hierarchical	Namespace Principal	Owner / cert hosts	Owner	✘	Owner
Trust Schema [80]	Hierarchical	Designated CA	Owner	Owner	✘	✘
NDNSSEC [74]	Hierarchical	DNS zone owner	Owner / DNS	Owner	RFC5011 [71] RFC7583 [53]	Issuer
IBC [87]	Basic	PKG	NA	Owner / PKG	✘	✘
HIBC [28]	Hierarchical	PKG	NA	Owner / PKG	✘	✘

of name ownership) or a namespace principal (*i.e.,* producer) must acquire a new identity and accordingly a new namespace. For Trust Schema, all allocations hold as long as the local schema is considered as valid. As such, trust rules are defined once and following modifications equal to defining a new Trust Schema. In addition, absence of methods to link Trust Schema with another prevents namespace delegations to other principals with their own set of Trust Schema. As the relation between a name and its owner is not intrinsic in any of the proposed approaches, name allocations are all secured by some form of cryptographic attestations. For (H)IBC, creating a private key from a namespace prefix (identity) is the proof of allocation by the PKG (or indirectly through NRS). For Trust Schema and NDN PKM, a trust chain that ends with a trust anchor (TA) serves to secure name allocations. And in NDNSSEC, all allocations are secured by DNSSEC.

Public Key Management. All but IBC rely on a hierarchical trust model. Beside being advantageous for scalability, a hierarchical trust model which constrains key management can also improve security. For example, in NDN PKM [79], having a CA (same as namespace principal) compromised does not have any negative impacts on other namespaces. In NDNSSEC [74] and NDN PKM the roles of namespace principal respectively DNS zone owner are the same as the CA for that namespace. Trust Schema [80] decouples these two roles and uses certificate names in trust rules to designate CAs for a set of names. In contrast to the other solutions, IBC and HIBC do not certify public keys but generate private keys. This poses a challenge to the non-repudiation property of signature as at least theoretically both PKG and key owner can generate signatures for the same identity. At the same time, public keys in (H)IBC do not require distribution and can be generated by relying parties on the fly using only public parameters of the (root) PKG. It is noteworthy that only NDNSSEC allow for key rollovers (although not discussed by the authors) by relying on DNSSEC methods of rollover [53, 71].

5 CONCLUSION

In this paper, we contributed a systematization of knowledge on namespace and public key management. We surveyed more than 30 NDN applications, proposed at ACM ICN 2015-2021 and related venues, to summarize their assumptions and requirements on naming and security. Lack of consensus on naming and namespace management among existing applications poses a potential cause for conflict when different applications with conflicting naming constraints should run in the same context, *e.g.,* in a global scope. Similarly, missing well-defined public key management, *e.g.,* to establish trust or to revoke compromised keys, poses a serious hurdle to achieve confidentiality, authentication, or access control that are based on public key cryptography. We analyzed current research that introduces approaches to implement naming and key management in NDN. Our comparison was structured along two taxonomies. We observed shortcomings in namespace management regarding authorization (constrained allocation), scalability (lack of delegability), and scope (limited governance policies and procedure).

Open challenges on public key management relate to simplifying bootstrapping procedures (as basis to establish trust relations) and providing procedures for key rollover and revocation (as part of key management). We hope that this paper serves as a systematic guide for both improving existing approaches and highlighting integral features and pitfalls that might arise when designing future solutions.

ACKNOWLEDGMENTS

We would like to thank our shepherd Lixia Zhang and the anonymous reviewers for their detailed and helpful feedback. This work was supported in parts by the German Federal Ministry of Education and Research (BMBF) within the projects Deutsches Internet-Institut (grant no. 16DII111) and PIVOT.

REFERENCES

[1] C. Adams, S. Farrell, T. Kause, and T. Mononen. 2005. *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP).* RFC 4210. IETF. http://tools.ietf.org/rfc/rfc4210.txt

[2] Sripriya S Adhatarao, Jiachen Chen, Mayutan Arumaithurai, Xiaoming Fu, and K K Ramakrishnan. 2016. Comparison of Naming Schema in ICN. In *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN '16).* IEEE, Piscataway, NJ, USA, 1–6.

[3] Alexander Afanasyev. 2013. *Addressing Operational Challenges in Named Data Networking Through NDNS Distributed Database.* Ph. D. Dissertation. University of California Los Angeles.

[4] Alexander Afanasyev, Xiaoke Jiang, Yingdi Yu, Jiewen Tan, Yumin Xia, Allison Mankin, and Lixia Zhang. 2017. NDNS: A DNS-Like Name Service for NDN. In *2017 26th International Conference on Computer Communication and Networks (ICCCN '17).* IEEE, Piscataway, NJ, USA, 1–9.

[5] Alexander Afanasyev, Cheng Yi, Lan Wang, Beichuan Zhang, and Lixia Zhang. 2015. SNAMP: Secure namespace mapping to scale NDN forwarding. In *2015 IEEE Conference on Computer Communications Workshops (INCOCOM).* IEEE, Piscataway, NJ, USA, 281–286.

[6] Marica Amadeo, Claudia Campolo, and Antonella Molinaro. 2014. Multi-Source Data Retrieval in IoT via Named Data Networking. In *Proceedings of the 1st ACM Conference on Information-Centric Networking* (Paris, France) *(ICN '14).* Association for Computing Machinery, New York, NY, USA, 67–76.

[7] Apple. 2022. *Apple Root Certificate Program.* Apple Inc. https://www.apple.com/certificateauthority/ca_program.html [Online; accessed: 2022-05-19].

[8] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. *DNS Security Introduction and Requirements.* RFC 4033. IETF. http://tools.ietf.org/rfc/rfc4033.txt

[9] Tohru Asami, Byambajav Namsraijav, Yoshihiko Kawahara, Kohei Sugiyama, Atsushi Tagami, Tomohiko Yagyu, Kenichi Nakamura, and Toru Hasegawa. 2015. Moderator-Controlled Information Sharing by Identity-Based Aggregate Signatures for Information Centric Networking. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking* (San Francisco, California, USA) *(ICN '15).* Association for Computing Machinery, New York, NY, USA, 157–166.

[10] Onur Ascigil, Sergi Reñé, George Xylomenos, Ioannis Psaras, and George Pavlou. 2017. A keyword-based ICN-IoT platform. In *Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN '17).* Association for Computing Machinery, New York, NY, USA, 22–28.

[11] Md. Bari, Shihabur Chowdhury, Reaz Ahmed, Raouf Boutaba, and Bertrand Mathieu. 2012. A Survey of Naming and Routing in Information-centric Networks. *IEEE Communications Magazine* 50, 12 (Dec. 2012), 44–53.

[12] Stefan Bechtold. 2003. Governance in Namespaces. *SSRN Electronic Journal* 36, 3 (2003), 82 pages.

[13] Matt Blaze, Joan Feigenbaum, and Jack Lacy. 1996. Decentralized Trust Management. In *Proceedings of 1996 IEEE Symposium on Security and Privacy.* IEEE, Piscataway, NJ, USA, 164–173.

[14] Ranit Chatterjee, Sushmita Ruj, and Sipra DasBit. 2020. Public Key Infrastructure for Named Data Networks. In *Proceedings of the 21st International Conference on Distributed Computing and Networking* (Kolkata, India) *(ICDCN '20).* Association for Computing Machinery, New York, NY, USA, 1–10.

[15] Jiachen Chen, Mayutan Arumaithurai, Xiaoming Fu, and K. K. Ramakrishnan. 2016. CNS: Content-oriented Notification Service for Managing Disasters. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking (ICN '16).* Association for Computing Machinery, New York, NY, USA, 122–131.

[16] Douglas E. Comer and Larry L. Peterson. 1989. Understanding naming in distributed systems. *Distributed Computing* 3, 2 (June 1989), 51–60.

[17] Alberto Compagno, Mauro Conti, and Ralph Droms. 2016. OnboardICNg: a Secure Protocol for On-boarding IoT Devices in ICN. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking (ICN '16).* Association for Computing Machinery, New York, NY, USA, 166–175.

[18] A. Cooper and J. Postel. 1992. *The US Domain.* RFC 1386. IETF. http://tools.ietf.org/rfc/rfc1386.txt

[19] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. 2008. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.* RFC 5280. IETF. http://tools.ietf.org/rfc/rfc5280.txt

[20] S. Farrell, D. Kutscher, C. Dannewitz, B. Ohlman, A. Keranen, and P. Hallam-Baker. 2013. *Naming Things with Hashes.* RFC 6920. IETF. http://tools.ietf.org/rfc/rfc6920.txt

[21] General Services Administration. 2022. *Federal Public Key Infrastructure Guide Introduction – Trust Stores.* https://github.com/GSA/ficam-playbooks/blob/88ad6fff953521fbd8ee762ece59c6be5db750ef/_fpki/3_fpki_truststores.md [Online; accessed: 2022-05-19].

[22] Ali Ghodsi, Teemu Koponen, Jarno Rajahalme, Pasi Sarolahti, and Scott Shenker. 2011. Naming in Content-Oriented Architectures. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking* (Toronto, ON, Canada) *(ICN '11).* ACM, New York, New York, USA, 1–6.

[23] Global Commission on Internet Governance. 2017. Who Runs the Internet?: The Global Multi-stakeholder Model of Internet Governance.

[24] Peter Gusev and Jeff Burke. 2015. NDN-RTC: Real-Time Videoconferencing over Named Data Networking. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking* (San Francisco, CA, USA) *(ICN '15).* Association for Computing Machinery, New York, NY, USA, 117–126.

[25] Cenk Gündoğan, Christian Amsüss, Thomas C. Schmidt, and Matthias Wählisch. 2021. Reliable Firmware Updates for the Information-Centric Internet of Things. In *Proceedings of the 8th ACM Conference on Information-Centric Networking (ICN '21).* Association for Computing Machinery, New York, NY, USA, 59–70.

[26] Cenk Gündoğan, Jakob Pfender, Michael Frey, Thomas C. Schmidt, Felix Shzu-Juraschek, and Matthias Wählisch. 2019. Gain More for Less: The Surprising Benefits of QoS Management in Constrained NDN Networks. In *Proceedings of the 6th ACM Conference on Information-Centric Networking (ICN '19).* Association for Computing Machinery, New York, NY, USA, 141–152.

[27] P. Hallam-Baker, R. Stradling, and J. Hoffman-Andrews. 2019. *DNS Certification Authority Authorization (CAA) Resource Record.* RFC 8659. IETF. http://tools.ietf.org/rfc/rfc8659.txt

[28] Balkis Hamdane, Rihab Boussada, Mohamed Elhoucine Elhdhili, and Sihem Guemara El Fatmi. 2017. Hierarchical Identity Based Cryptography for Security and Trust in Named Data Networking. In *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '17).* IEEE, Piscataway, NJ, USA, 226–231.

[29] Ehsan Hemmati and J.J. Garcia-Luna-Aceves. 2015. A New Approach to Name-Based Link-State Routing for Information-Centric Networks. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking* (San Francisco, CA, USA) *(ICN '15).* Association for Computing Machinery, New York, NY, USA, 29–38.

[30] P. Hoffman and J. Schlyter. 2012. *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA.* RFC 6698. IETF. http://tools.ietf.org/rfc/rfc6698.txt

[31] A K M Mahmudul Hoque, Syed Obaid Amin, Adam Alyyan, Beichuan Zhang, Lixia Zhang, and Lan Wang. 2013. NLSR: Named-Data Link State Routing Protocol. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking* (Hong Kong, China) *(ICN '13).* Association for Computing Machinery, New York, NY, USA, 15–20.

[32] Saddam Hussain, Syed Sajid Ullah, Abdu Gumaei, Mabrook Al-Rakhami, Ijaz Ahmad, and Syed Muhammad Arif. 2021. A Novel Efficient Certificateless Signature Scheme for the Prevention of Content Poisoning Attack in Named Data Networking-Based Internet of Things. *IEEE Access* 9 (2021), 40198–40215.

[33] International Organization for Standardization. 2019. *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services.* Standard. International Telecommunication Union.

[34] International Organization for Standardization. 2020. *Information technology – Open Systems Interconnection – The Directory – Part 1: Overview of concepts, models and services.* Standard ISO/IEC 9594-1:2020. International Organization for Standardization, Geneva, CH.

[35] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. 2009. Networking Named Content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies* (Rome, Italy) *(CoNEXT '09).* Association for Computing Machinery, New York, NY, USA, 1–12.

[36] Mike Just. 2011. *PKI Trust Models.* Springer US, Boston, MA, 936–939.

[37] R. Khare. 1999. What's in a Name? Trust. Internet-Scale Namespaces, Part II. *IEEE Internet Computing* 3, 6 (1999), 80–84.

[38] Siham Khoussi, Davide Pesavento, Lotfi Benmohamed, and Abdella Battou. 2017. NDN-Trace: A Path Tracing Utility for Named Data Networking. In *Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN '17).* Association for Computing Machinery, New York, NY, USA, 116–122.

[39] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. 2007. A data-oriented (and beyond) network architecture. *ACM SIGCOMM Computer Communication Review* 37, 4 (Oct. 2007), 181.

[40] Michał Król, Karim Habak, David Oran, Dirk Kutscher, and Ioannis Psaras. 2018. RICE: Remote Method Invocation in ICN. In *Proceedings of the 5th ACM Conference on Information-Centric Networking (ICN '18).* Association for Computing Machinery, New York, NY, USA, 1–11.

[41] Michał Król, Spyridon Mastorakis, David Oran, and Dirk Kutscher. 2019. Compute First Networking: Distributed Computing meets ICN. In *Proceedings of the 6th ACM Conference on Information-Centric Networking (ICN '19).* Association for Computing Machinery, New York, NY, USA, 67–77.

[42] Michał Król and Ioannis Psaras. 2017. NFaaS: Named Function as a Service. In *Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN '17).* Association for Computing Machinery, New York, NY, USA, 134–144.

[43] Jun Kurihara, Kenji Yokota, and Atsushi Tagami. 2016. A Consumer-Driven Access Control Approach to Censorship Circumvention in Content-Centric Networking. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking (ICN '16).* Association for Computing Machinery, New York, NY, USA,

186–194.
[44] Dirk Kutscher, Laura Al Wardani, and T M Rayhan Gias. 2021. Vision: information-centric dataflow. In *Proceedings of the 8th ACM Conference on Information-Centric Networking (ICN '21)*. Association for Computing Machinery, New York, NY, USA, 52–58.
[45] Teng Liang, Zhongda Xia, Guoming Tang, Yu Zhang, and Beichuan Zhang. 2021. NDN in Large LEO Satellite Constellations: A Case of Consumer Mobility Support. In *Proceedings of the 8th ACM Conference on Information-Centric Networking* (Paris, France) *(ICN '21)*. Association for Computing Machinery, New York, NY, USA, 1–12.
[46] Steve Lloyd and Carlisle Adams. 2011. *Key Management*. Springer US, Boston, MA, 683–688.
[47] Priya Mahadevan, Ersin Uzun, Spencer Sevilla, and J.J. Garcia-Luna-Aceves. 2014. CCN-KRS: A Key Resolution Service for CCN. In *Proceedings of the 1st International Conference on Information-centric Networking (ICN '14)*. Association for Computing Machinery, New York, New York, USA, 97–106.
[48] Elisa Mannes and Carlos Maziero. 2019. Naming content on the network layer: A security analysis of the information-centric network model. *Comput. Surveys* 52, 3 (2019), 1–28.
[49] Dima Mansour and Christian Tschudin. 2016. Towards a Monitoring Protocol Over Information-Centric Networks. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking (ICN '17)*. Association for Computing Machinery, New York, NY, USA, 60–64.
[50] P.V. Mockapetris. 1987. *Domain names - concepts and facilities*. RFC 1034. IETF. http://tools.ietf.org/rfc/rfc1034.txt
[51] Ilya Moiseenko and Dave Oran. 2016. TCP/ICN: Carrying TCP over Content Centric and Named Data Networks. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking (ICN '17)*. Association for Computing Machinery, New York, NY, USA, 112–121.
[52] Philipp Moll, Sebastian Theuermann, Natascha Rauscher, Hermann Hellwagner, and Jeff Burke. 2019. Inter-Server Game State Synchronization Using Named Data Networking. In *Proceedings of the 6th ACM Conference on Information-Centric Networking* (Macao, China) *(ICN '19)*. Association for Computing Machinery, New York, NY, USA, 12–18.
[53] S. Morris, J. Ihren, J. Dickinson, and W. Mekking. 2015. *DNSSEC Key Rollover Timing Considerations*. RFC 7583. IETF. http://tools.ietf.org/rfc/rfc7583.txt
[54] Mozilla. 2022. *Common CA Database*. Mozilla Foundation. https://www.ccadb.org [Online; accessed: 2022-05-19].
[55] Xavier Mwangi and Karen Sollins. 2018. MNDN: Scalable Mobility Support in Named Data Networking. In *Proceedings of the 5th ACM Conference on Information-Centric Networking (ICN '18)*. Association for Computing Machinery, New York, NY, USA, 117–124.
[56] Named Data Networking Project. 2021. *NDN Certificate Format Version 2.0*. NDN Project Team. https://github.com/named-data/ndn-cxx/blob/4999b2eddd7da10bc934a27f2f151e37f2095765/docs/specs/certificate.rst Online; accessed 2022-05-03.
[57] Named Data Networking Project. 2021. *NDN Packet Format Specification 0.3 – KeyLocator*. NDN Project Team. https://github.com/named-data/NDN-packet-spec/blob/acaccafd710a306df405e4b40302080d729b4ade/signature.rst#keylocator Online; accessed 2022-05-03.
[58] Named Data Networking Project. 2021. *NDN Packet Format Specification 0.3 – NDN Name Format*. NDN Project Team. https://github.com/named-data/NDN-packet-spec/blob/23e340cb138afa95ee228dffb8c50870eea26633/name.rst#ndn-name-format Online; accessed 2022-05-03.
[59] NDN Project Team. 2020. *NFD – Control Command*. https://redmine.named-data.net/projects/nfd/wiki/ControlCommand/52 [Online; accessed: 2022-05-17].
[60] NDN Project Team. 2021. *NDN Technical Memo: Naming Conventions*. techreport NDN-0022.
[61] Kathleen Nichols. 2019. Lessons Learned Building a Secure Network Measurement Framework Using Basic NDN. In *Proceedings of the 6th ACM Conference on Information-Centric Networking* (Macao, China) *(ICN '19)*. Association for Computing Machinery, New York, NY, USA, 112–122.
[62] Eric Osterweil, Pouyan Fotouhi Tehrani, Thomas C. Schmidt, and Matthias Wählisch. 2022. From the Beginning: Key Transitions in the First 15 Years of DNSSEC. *IEEE Transactions on Network and Service Management* (2022). [Early Access].
[63] Sanjeev Kaushik Ramani, Reza Tourani, George Torres, Satyajayant Misra, and Alexander Afanasyev. 2019. NDN-ABS: Attribute-Based Signature Scheme for Named Data Networking. In *Proceedings of the 6th ACM Conference on Information-Centric Networking (ICN '19)*. Association for Computing Machinery, New York, NY, USA, 123–133.
[64] Daniel Rezende, Carlos Maziero, and Elisa Mannes. 2018. A distributed online certificate status protocol for named data networks. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. Association for Computing Machinery, New York, NY, USA, 2102–2108.
[65] Ronald L. Rivest and Butler Lampson. 1996. SDSI – A Simple Distributed Security Infrastructure.
[66] J H Saltzer. 1978. Naming and Binding of Objects. In *Operating Systems - An Advanced Course*. Springer, Berlin, Heidelberg, 99–208.
[67] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. 2013. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. RFC 6960. IETF. http://tools.ietf.org/rfc/rfc6960.txt
[68] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C. Schmidt, and Matthias Wählisch. 2018. The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem. In *Proceedings of the Internet Measurement Conference 2018 (IMC '18, October 2016)*. Association for Computing Machinery, New York, NY, USA, 343–349.
[69] Adi Shamir. 1985. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology*, George Robert Blakley and David Chaum (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 47–53.
[70] Karen R. Sollins. 2008. *Recursively invoking Linnaeus: A Taxonomy for Naming Systems*. Technical Report MIT-CSAIL-TR-2008-064. MIT.
[71] M. StJohns. 2007. *Automated Updates of DNS Security (DNSSEC) Trust Anchors*. RFC 5011. IETF. http://tools.ietf.org/rfc/rfc5011.txt
[72] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. 2002. Internet indirection infrastructure. *ACM SIGCOMM Computer Communication Review* 32, 4 (Oct. 2002), 73.
[73] Pouyan Fotouhi Tehrani, Luca Keidel, Eric Osterweil, Jochen H. Schiller, Thomas C. Schmidt, and Matthias Wählisch. 2019. NDNSSEC: Namespace Management in NDN with DNSSEC. In *Proceedings of the 6th ACM Conference on Information-Centric Networking* (Macao, China) *(ICN '19)*. Association for Computing Machinery, New York, NY, USA, 171–172.
[74] Pouyan Fotouhi Tehrani, Eric Osterweil, Jochen H. Schiller, Thomas C. Schmidt, and Matthias Wählisch. 2019. The Missing Piece: On Namespace Management in NDN and How DNSSEC Might Help. In *Proceedings of the 6th ACM Conference on Information-Centric Networking* (Macao, China) *(ICN '19)*. Association for Computing Machinery, New York, NY, USA, 37–43.
[75] Jeff Thompson, Peter Gusev, and Jeff Burke. 2019. NDN-CNL: A Hierarchical Namespace API for Named Data Networking. In *Proceedings of the 6th ACM Conference on Information-Centric Networking* (Macao, China) *(ICN '19)*. Association for Computing Machinery, New York, NY, USA, 30–36.
[76] Zheng Wang and Liyuan Xiao. 2014. Emergency Key Rollover in DNSSEC. In *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE Computer Society, Los Alamitos, CA, USA, 598–604.
[77] Wang Yang, Fan Wu, and Kaijin Tian. 2021. High performance adaptive video streaming using NDN WLAN multicast. In *Proceedings of the 8th ACM Conference on Information-Centric Networking* (2021-09) *(ICN '21)*. Association for Computing Machinery, New York, NY, USA, 42–51.
[78] Tianyuan Yu, Hongcheng Xie, Siqi Liu, Xinyu Ma, Xiaohua Jia, and Lixia Zhang. 2022. CertRevoke: A Certificate Revocation Framework for Named Data Networking. In *Proceedings of the 9th ACM Conference on Information-Centric Networking* (Osaka, Japan) *(ICN '22)*. Association for Computing Machinery, New York, NY, USA. [Early Access].
[79] Yingdi Yu. 2015. *Public Key Management in Named Data Networking*. techreport NDN-0029. UCLA.
[80] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. 2015. Schematizing Trust in Named Data Networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking* (San Francisco, California, USA) *(ICN '15)*. Association for Computing Machinery, New York, New York, USA, 177–186.
[81] Yingdi Yu, Alexander Afanasyev, Jan Seedorf, Zhiyi Zhang, and Lixia Zhang. 2017. NDN DeLorean: An Authentication System for Data Archives in Named Data Networking. In *Proceedings of the 4th ACM Conference on Information-Centric Networking* (Berlin, Germany) *(ICN '17)*. Association for Computing Machinery, New York, NY, USA, 11–21.
[82] Haitao Zhang. 2018. *NDNFit: An Open mHealth Application Built on Named Data Networking*. Ph. D. Dissertation. University of California Los Angeles.
[83] Haitao Zhang, Zhehao Wang, Christopher Scherb, Claudio Marxer, Jeff Burke, Lixia Zhang, and Christian Tschudin. 2016. Sharing MHealth Data via Named Data Networking. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking* (Kyoto, Japan) *(ICN '16)*. Association for Computing Machinery, New York, NY, USA, 142–147.
[84] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Kc Claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. 2014. Named data networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (July 2014), 66–73.
[85] Lixia Zhang, Deborah Estrin, Burke Jeffrey, Van Jacobso, James D. Thornton, Diana K. Smetters, Beichuan Zhang, Gene Tsudik, kc claffy, Dmitri Krioukov, Dan Massey, Christos Papadopoulos, Tarek Abdelzaher, Lan Wang, Patrick Crowley, and Edmund Yeh. 2010. *Named Data Networking (NDN) Project*. Technical Report NDN-0001.
[86] Minsheng Zhang, Vince Lehman, and Lan Wang. 2017. Scalable name-based data synchronization for named data networking. In *IEEE INFOCOM 2017 - IEEE*

Conference on Computer Communications. IEEE, Piscataway, NJ, USA, 9 pages.
[87] Xinwen Zhang, Katharine Chang, Huijun Xiong, Yonggang Wen, Guangyu Shi, and Guoqiang Wang. 2011. Towards name-based trust and security for content-centric network. In *2011 19th IEEE International Conference on Network Protocols*. IEEE, Piscataway, NJ, USA, 1–6.
[88] Yu Zhang, Zhongda Xia, Spyridon Mastorakis, and Lixia Zhang. 2018. KITE: Producer Mobility Support in Named Data Networking. In *Proceedings of the 5th ACM Conference on Information-Centric Networking (ICN '18)*. Association for Computing Machinery, New York, NY, USA, 125–136.
[89] Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Spyridon Mastorakis, Yanbiao Li, Alexander Afanasyev, and Lixia Zhang. 2018. An Overview of Security Support in Named Data Networking. *IEEE Communications Magazine* 56, 11 (Nov. 2018), 62–68.
[90] Zhenkai Zhu and Alexander Afanasyev. 2013. Let's ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking. In *2013 21st IEEE International Conference on Network Protocols (ICNP '13)*. IEEE, Piscataway, NJ, USA, 1–10.

A ADDITIONAL BACKGROUND ON DNS, DNSSEC, AND WEB PKI

A.1 DNS and DNSSEC

The domain namespace [50] is managed in terms of *zones*, *i.e.,* subtrees within the DNS hierarchy, and a zone administrator can *delegate* any branch under its sub-tree to another authority. For example, `ietf.org.` ("IETF Trust") has been delegated by the management authority of the zone `org.` ("Public Interest Registry"). DNSSEC [8], the DNS security extensions, introduce data origin authentication, data integrity, and authenticated denial of existence to DNS. DNSSEC secures domain names by binding special RRs that allow for cryptographic verification to domain names. These include public keys (`DNSKEY`), digital signatures (`RRSIG`), and delegation signers (`DS`) to reflect the DNS hierarchy.

DNSSEC keys are used to sign set of RRs of the same type (*RRsets*) instead of single RRs; and the same RRset can be signed by different keys at the same time. When a relying party recursively resolves a name, the RP validates a zone as *secure* if (*i*) the RRsets are signed by at least one valid key included in the `DNSKEY` RRset, and (*ii*) a *chain of trust* can be established from a trust anchor (commonly the root zone) to that zone. The root zone uses a *well-known* self-signed key signing key (KSK) to sign its `DNSKEY`s and uses its zone signing key (ZSK) to sign `DS` records for top-level domains (TLD) such as `.com`. Similarly, when a second-level domain name (SLD) is delegated from a TLD to a DNSSEC-enabled zone, at least one `DS` record is created at the TLD zone upon the request of the delegated zone authority. This iterative procedure creates unique verifiable path from the root zone to any arbitrary DNSSEC-enabled zone. Although there are no constraints for an RP on designating arbitrary zones as trust anchors, in practice RPs use the root zone as the single trusted third party in DNSSEC PKI. A zone can only delegate sections of DNS namespace under its subspace, a zone owner can only certify keys of its immediate child zones.

A.2 Web PKI

Based on X.509 certificates [19], Web PKI enables identification and authentication of service endpoints over the web. Here, X.509 certificates are used to bind a *subject*, a real-world identity, to a public key. Originally, a subject was intended to be represented through a *distinguished name*, a unique entry in a global hierarchical directory [33]. Such global directory, however, never came into existence, making distinguished names prone to collision and not suitable for unique authentication or identification of the certificate holder. This shortcoming is not an issue in Web PKI as X.509 certificates are also bound to domain names (as globally unique identifiers). Here, Certification Authorities (CA) certify bindings in a certificate through their signature. A CA can also delegate its authority to *intermediate* CAs, which issues a subscriber a certificate. To validate a certificate, RPs, *i.e.,* web clients, must first acquire certificates for a set of trustworthy CAs through an out-of-band channel. These certificates form the so-called *trust store* and are maintained by browser vendors, operating system companies, *etc.* and not RPs themselves. In general, any trusted CA can certify a public key for any arbitrary subscriber and a single subscriber can be certified by multiple CAs; consequently and in contrast to DNSSEC, there is not necessarily one unique verifiable paths from a given subject or domain name to a CA.

Various solutions have been proposed to remove the ambiguity regarding public keys and responsible CAs for a specific domain name. DNS Certification Authority Authorization (`CAA` RR) [27], for example, allows a domain name owner to explicitly specify which CAs are allowed to issue a certificate for that domain name. Alternatively, the DNS-Based Authentication of Named Entities (DANE) allows describing certificates that are expected to be provided a service endpoint (using `TLSA` RRs [30]). DANE practically extends a DNSSEC trust chain to Web PKI. Finally, an RP validates a certificate as long as there is signature chain from that certificate to a certificate in its trust store.