Universidade do Minho U.PORTO
Universidade de Aveiro
2019
Departamento de Eletrónica,
Telecomunicações e Informática

Rui Abreu de Carvalho Ferreira

Segurança e Privacidade em Terminologia de Rede

Security and Privacy in Network Namespaces

Universidade de Aveiro
2019
Departamento de Eletrónica,
Telecomunicações e Informática

Rui Abreu de Carvalho Ferreira

Segurança e Privacidade em Terminologia de Rede

Security and Privacy in Network Namespaces

Tese apresentada às Universidades de Aveiro Minho e Porto para cumprimento dos requisitos necessários à obtenção do grau de Doutor no âmbito do programa doutoral MAP-i, realizada sob a orientação científica do Doutor Rui Aguiar, Professor catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Este trabalho é dedicado aos meus pais. Pelos milhares de pequenas coisas que fazem toda a diferença. Acima de tudo por me ensinarem a curiosidade, o único requisito verdadeiramente necessário.

Obrigado.

o júri / the jury

presidente / president

Doutora Maria Hermínia Deulonder Correia Amado Laurel
Professora Catedrática
Departamento de Línguas e Culturas
Universidade de Aveiro

vogais / examiners committee

Doutor Jorge Miguel Matos Sousa Pinto
Professor Associado com Agregação
Departamento de Informática
Universidade do Minho

Doutor Miguel Nuno Dias Alves Pupo Correia
Professor Associado
Instituto Superior Técnico
Universidade de Lisboa

Doutor João Paulo da Silva Machado Garcia Vilela
Professor Auxiliar
Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Doutor Amaro Fernandes de Sousa
Professor Auxiliar
Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

Doutor Rui Luis Andrade Aguiar (orientador)
Professor Catedrático
Departmento de Electrónica Telecomunicações e Informàtica
Universidade de Aveiro

agradecimentos /
acknowledgements

Ao Professor Rui Aguiar por todo o apoio e motivação que me ajudaram a atingir este objectivo. Por todas as contribuições, ideias, o constante desafio intelectual e sobretudo pela oportunidade que me deu para trabalhar em ciência.

A todos os meus colegas do grupo de redes de telecomunicações do Instituto de Telecomunicações de Aveiro, pelas muitas horas que passamos juntos durante estes anos, e sem os quais este trabalho não seria possivel. E também a todos os colegas investigadores que concordando ou discordando contribuiram para a discussão nestas matérias.

Por fim uma palavra especial de apreço para o Alfredo Matos, Rodolphe Marques, e José Quevedo, que recordo com apreço por dizerem sempre o que pensam, mas pensarem sempre no que dizem.

Palavras Chave segurança, privacidade, redes, identificadores, protocolos.

Resumo

Segurança e Privacidade são dois topicos que marcam a agenda na discussão sobre a sociedade digital. Um aspecto particularmente subtil nesta discussão é a forma como atribuímos nomes a recursos na rede, uma escolha com consequências práticas no funcionamento dos diferentes protocols de rede, na forma como se implementam diferentes mecanismos de segurança e na privacidade das várias partes envolvidas. Este problema torna-se ainda mais significativo quando se considera que, para promover a interoperabilidade entre diferentes redes, mecanismos autónomos tornam esta informação acessível em contextos que vão para lá do que era pretendido.

Esta tese foca-se nas consequências de diferentes políticas de atribuição de nomes no contexto de diferentes protocols de rede, para efeitos de segurança e privacidade. Com base no estudo deste problema, são propostas soluções que, através de diferentes políticas de atribuição de nomes, permitem introduzir mecanismos de segurança adicionais ou mitigar problemas de privacidade em diferentes protocolos. Isto resulta na implementação de mecanismos de segurança sobre protocolos de descoberta inseguros, assim como na introdução de mecanismos de atribuiçao e resolução de nomes que se focam na protecçao da privacidade.

O principal veículo para a implementação destas soluções é através de serviços e protocolos de rede de uso geral. No entanto, a aplicabilidade destas soluções extende-se também a outros tópicos de investigação que recorrem a mecanismos de resolução de nomes para implementar soluções de interoperabilidade, nomedamente a Internet das Coisas (IoT) e redes centradas na informação (ICN).

Keywords naming, security, privacy, networks, identifiers, protocols.

Abstract

Security and Privacy are now at the forefront of modern concerns, and drive a significant part of the debate on digital society. One particular aspect that holds significant bearing in these two topics is the naming of resources in the network, because it directly impacts how networks work, but also affects how security mechanisms are implemented and what are the privacy implications of metadata disclosure. This issue is further exacerbated by interoperability mechanisms that imply this information is increasingly available regardless of the intended scope.

This work focuses on the implications of naming with regards to security and privacy in namespaces used in network protocols. In particular on the implementation of solutions that provide additional security through naming policies or increase privacy. To achieve this, different techniques are used to either embed security information in existing namespaces or to minimise privacy exposure. The former allows bootstraping secure transport protocols on top of insecure discovery protocols, while the later introduces privacy policies as part of name assignment and resolution.

The main vehicle for implementation of these solutions are general purpose protocols and services, however there is a strong parallel with ongoing research topics that leverage name resolution systems for interoperability such as the Internet of Things (IoT) and Information Centric Networks (ICN), where these approaches are also applicable.

Contents

List of Figures

List of Tables

Acronyms

4WARD	Architecture and design for the future Internet
API	Application Programming Interface
AS	Autonomous System
BLE	Bluetooth Low Energy
CA	Certificate Authority
CCN	Content Centric Networks
CDN	Content Distribution Network
CoaP	Constrained Application Protocol
DAG	Directed Acyclic Graph
DHT	Distributed Hash Table
DLNA	Digital Living Network Alliance
DNS	Domain Name System
DNS-SD	DNS-based Service Discovery
DNSSEC	DNS Security
DONA	Data Oriented Network Architecture
DoS	Denial of Service Attack
EDNS	Extension mechanisms for DNS
EID	Entity Identifier
ELOC	Entity Locator
FP7	7th Framework Program
GHR	Global Handle Registry
GNS	GNU Name System
GPS	Global Positioning System
HIP	Host Identity Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
i3	Internet Indirection Infrastructure
ICANN	Internet Corporation for Assigned Names and Numbers
ICN	Information Centric Networks
IdM	Identity Management
IETF	Internet Engineering Task Force
IM	Instant Messaging
IoT	Internet of Things
IP	Internet Protocol
IPFS	InterPlanetary File System
IPSec	IP Security Protocol
ITU-T	International Telecommunications Union - Telecom
L2CAP	Logical link control and adaptation protocol
LHS	Local Handle Services
LMP	Link Manager Protocol
MAC	Media Access Control
MIP	Mobile IP
MITM	Man-in-The-Middle
MQTT	Message Queuing Telemetry Transport
NDN	Named Data Networking
Netinf	Network of Information
NFC	Near Field Communication
NPSN	Named Publish Subscribe Networking
NRS	Name Resolution System
OASIS	Organization for the Advancement of Structured Information Standards
OSI	Open Systems Interconnection
P2P	Peer to Peer
PLA	Packet Level Authentication

PKI	Public Key Infrastructure
PMIP	Proxy Mobile IP
PSIRP	Publish Subscribe Internet Routing Paradigm
PURSUIT	Publish Subscribe Internet Technology
QoS	Quality of Service
QR Code	Quick Response Code
REST	Representational State Transfer
RFCOMM	Radio Frequency Communication
RFID	Radio-frequency identification
SAIL	Scalable and Adaptive Internet Solutions
SAML	Security Assertion Markup Language
SBN	Session Bound Namespace
SDP	Service Discovery Protocol
SFR	Semantic Free Referencing
SMTP	Simple Mail Transfer Protocol
SOP	Same Origin Policy
SP	Service Provider
SRVID	Service Type Identifier
SSDP	Simple Service Discovery Protocol
SWIFT	Secure Widespread Identity for Federated Telecommunications
TCP	Transmission Control Protocol
TLD	Top Level Domain
TLS	Transport Layer Security
ToR	The Onion Router
ToS	Terms of Service
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USN	Unique Service Name
UUID	Universally Unique Identifier
VDM	Virtual Device Manager
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WWW	World Wide Web
XEP	XMPP Extension Protocol
XIA	eXpressive Internet Architecture
XMPP	Extensible Messaging and Presence Protocol
XRDS	Extensible Resource Descriptor Sequence
XRI	Extensible Resource Identifier

Chapter 1

Introduction

Name (verb) - to give a name to (someone or something); to say the name of (someone or something); to choose (someone) to be (something)

Merriam-Webster Dictionary

As the first chapter, the introduction draws the background for the pursuit of a PhD in this topic. It defines the goals for this work, summarises the main contributions that it produced and pins the overall structure of this document.

1.1 Background & Motivation

The transitive verb *Name* holds three distinct meanings: the assignment of an identifying name to an entity, the use of the afore mentioned identifier to refer to an entity, and the appointment of functions based on the assignment of a name. Names are one of the key aspects of human communication. They work as pointers supported by language. In the absence of language, the natural substitute would be to point at things to refer to them. But hand waiving can only take us so far, and one would be hard pressed to point at faraway or conceptual entities. Fortunately, language allows us to name entities even if they are metaphysical, geographically distant, or no longer exist at all.

In distributed systems[1], naming is the starting point for establishing identity for entities in a system. Because any sufficiently complex system needs to distinguish between multiple involved entities, the bootstrapping of a system requires discovering existing nodes assigning them names. This holds true for many types of computer

systems, processes running inside a CPU, or files in a hard drive. Computer nodes in a network are no exception: they are assigned names as they join the network, and these assignments play a key role in their later operations as names are resolved into other objects. In modern network stacks, different layers in the stack use different namespaces. The purpose of name resolution is to enable discovery, resolution and assignment functions around these namespaces. In essence, to dereference a name into an object.

The topic of naming in computer networks is particularly interesting because it easily breaks across the rigid boundaries of layered models. We reason about network internals as independent layers in the OSI or IP stacks, but naming design decisions can (and do) bleed out of their intended layer, sometimes in subtle ways that change the behaviour of the network as a whole. Name resolution systems are often used to facilitate the interconnection between layers, mapping between different layers, or as a design kludge to collapse abstraction layers. Since the number of nodes and services in a network at any given time is an unknown, this implies relying on external entities as a source of knowledge, and as the boundaries of the network change it can result on the reliance of transitive relations of trust.

This thesis is put forward as a study on the role of *Name Resolution* functions, and its applications going forward in the face of changes to the architectural design of computer networks. The goal is to improve the understanding of what are the future requirements that drive this type of system, how they clash with the existing architecture(s), and how to address them. In particular, this thesis, highlights compromises in security and privacy that surround name resolution function composition.

The focus of this work are solutions that build upon existing architectures and protocols, following an evolutionary approach, rather than a clean slate design. Over the years, multiple industry and research initiatives have looked at designing a new name system for the Internet [2, 3, 4, 5]. Despite significant technical and scientific achievements, not all of these contributions have reached notoriety, or are directly applicable in computer networks. As a rule of thumb, pragmatism is favoured over completeness, because Computer Networks are built as a decentralised effort - solutions that work are quickly integrated, until convergence occurs or a better solution presents itself.

My personal motivation for this work stems from a previously existing overlap in different research interests. Prior to starting this PhD, I was entangled with two research topics that deal closely with names in computer networks: the first was Privacy disclosure through network identifiers and the second Identity Management (IdM).

Privacy deals with the management of identifiers as means for controlling the disclosure of information, and my main goal in this context was to timely create or destroy

identifiers to establish efficient pseudonymity with regards to user expectations. Unsurprisingly, the network stack knows little about human expectations of privacy, and I realised that layered models can easily disrupt user assumptions.

Identity Management is mostly concerned with the assignment of Names, asserting proof of ownership over them (i.e. authentication) and on building services on top of this capability. Implicitly Identity Management requires strong properties from the underlying name system not only in terms of security but also for flexibility. A name is resolved differently based on contextual information, and names can refer to people, objects, or computers alike. The notion of ownership can become fuzzy and the process of assertion can differ for each type of entity, but ultimately it must take place as a series of network operations.

While I grasped on the intricacies of IdM privacy, in other fields, ongoing research topics challenge the structuring of the network stack design, making it worthwhile to review assumptions on the role name resolution in the network and the implications of these changes.

- Security & Privacy are now under the limelight, after recent events exposed the fragility of the Internet with regards to the naming and trust infrastructure that enable a variety of Man-in-The-Middle (MITM) and privacy attacks.
- The Internet of Things (IoT) proposes a vision where all devices are reachable, using an heterogeneous mixture of protocols. How to reach them and whether general expectations of security and privacy hold is an open research topic.
- Information Centric Networks (ICN) are a paradigm shift that reorganises the network stack to address content rather than endpoints. This shift in functionality implies changing how resources are named in the network, and clashes with notions of addressing locality.
- Naming services derive their value from from mapping names to objects, with one particular case being to map names to other names. The World Wide Web (WWW) is filled with such services, e.g. make Uniform Resource Locators (URLs) shorter, or provide obfuscation. These can be seen as interoperability enablers, and are a common approach to the composition of services or networks.

Interestingly all these topics touch on what *Name Resolution* functions are available in the network, and intersect in several points. IoT often assumes decentralised or ad-hoc scenarios, where device and service discovery are the only form of resolution available. Despite being decentralised, aspects of security and trust relations still need to hold, and are carried over from other environments. The paradigm in ICN is to address content using meaningful context, but content is named by its creators not its

carriers. As names defined at the application layer find their way into the lower layers of the network, a privacy concern is that content names hold far more information than traditional network addresses.

However the previously mentioned research topics operate in distinct areas, or target distinct purposes, and aligning goals across such a wide spectrum is not always possible. As professionals in this field, we are called upon to build effective bridges between all these aspects and technologies. As they become interconnected (as all networks do), concepts from one will inevitably leak into the others. Convergence efforts are already bridging these different requirements. ICN protocols [6, 7] use application names at the network layer, and are also targeting IoT [8, 9] scenarios. Global connectivity in IoT environments is achieved through mapping gateways that provide connectivity between networks through resolution and adaptation services.

All these factors, as well as an interest in related problems, led to my interest in pursuing a PhD in this area.

1.2 Naming Hurdles

Since identification is the foremost problem to solve in distributed systems, it comes as no surprise that entities in any system must be identifiable to enable effective communication.

There are compelling arguments to the establishment of a global namespace with security properties. In practice many of the current computer networks adopt this type of approach, because:

- consistent naming facilitates mobility across namespaces or protocols
- security functions are a fundamental requirement for globally connected systems
- global uniqueness facilitates the construction of a global network

The problem of consistent identification in IP networks is commonly referred to as a compromise between identification and network topology. There is extensive previous work that introduces topology independent and secure namespaces on top of existing network protocols, most notably the Host Identity Protocol (HIP)[10]. Two fundamental concepts support this protocol, first the generation of identifiers as a cryptographic hash of a key, and second the use of a consistent identifier over time that does not change as the node moves. This facilitates the introduction of two features over HIP: first the bootstrapping of secure communication protocols, such as IPSec, and second end host mobility mechanisms [11] that do not rely on the network provider. Many others exist, either built using name services for identifier translation [12, 13], or as overlay networks [14].

Global namespaces with security properties often compromises privacy for uniqueness and security, because names become unique across a larger scope and often include additional extractable information. In other words, the use of a single consistent identifier makes it easier for network entities to track nodes as they move across the network. To mitigate this problem one needs to introduce additional privacy controls. One example is the use of network layer pseudonymity[15] for Media Access Control (MAC) and Internet Protocol (IP) addresses. Pseudonymity refers to the transient assignment of names for specific roles or purposes, in this particular case the creation of names that limit privacy disclosure.

Not withstanding these improvements, two main gaps remain. First, as networks become interconnected it is unclear how these novel namespaces derive names from old namespaces, and how security and privacy semantics can be applied here, since pseudonymity approaches seen in existing network protocols do not apply directly to other types of namespaces (such as ICN or Content Centric Networks (CCN)). Second, some of the networks being interconnected lack solutions similar to HIP that would enable binding of secure names, and facilitate convergence using a global secure namespace. These are the topic of study for this PhD, as they become increasingly relevant with current convergence trends in present and future networks.

1.2.1 Leaking identifiers in network protocols

The value of privacy, with regards to network identifiers, is hard to assess a-priori without additional context about thei information meaning, assignment scope and duration. Many network protocols would not work without disclosure of network identifiers Hypertext Transfer Protocol (HTTP) is one such protocol. Since it is based on a point to point communication model, it requires exchanging identification about relevant endpoints. Other types of identifier based information disclosure are a side-effect of non communication features. For example, email [16] includes user IP addresses within protocol messages for auditing purposes, while HTTP [17] leaks the URL for the previously visited web page through the referer header.

Since some identifiers can be created through composition of existing identifiers, they can leak unnecessary information. One particular example of this happened with IPv6 address auto configuration [18] that includes the MAC address unique to each network card in the local part of the IP address. Thus IPv6 addresses generated this way could be used to identify each device, as the local part of the address never changes. A more privacy preserving mechanism is [19] which randomises the local part of the address, increasing address reuse over time.

Other types of privacy issues may arise from identifier leakage, depending on what kind of information can be extracted, or cross referenced from these identifiers. Some

protocols are designed to reduce privacy issues, and some service providers take additional measures to preserve user privacy, scrubbing unique identifiers from ancillary data and acting as an intermediary.

While network addresses do not seem to reveal much information, they provide unique identification of resources within a scope which is enough for collusion of information. A very common example is the mapping of network addresses onto geographic areas through existing geolocation databases [20] and from public IP registration information[21]. The most common method is to determine geographical location from IP address, but databases for other network identifiers such as stationary MAC addresses and wireless network names are also available.

Construction of these databases benefit from information gathered from multiple sources:

- the regional registries that assign IP networks to companies.
- mining of addresses in open networks, including additional information, such as the addresses of wireless and cellular access points, used to do reverse lookups.
- cross reference device network addresses with Global Positioning System (GPS) coordinates e.g. by using voluntary data from mobile devices.

While there is no foolproof way to associate an IP address with a location, some of these databases can accurately associate an IP with a specific street, or even a building [22].

This problem arises because these identifiers can be uniquely associated with an entity over a significant scope, and data mining of associated information is now a feasible operation. Since the association of an IP address to a location changes infrequently (in particular for blocks assigned to internet service providers) it is often not possible to simply change network addresses in a way that prevents the use of these databases. It is also important to understand that disclosure from one user has an impact in the surrounding users. Because addresses are organised in blocks, topological proximity usually implies geographical proximity, at least within regional blocks and provided there are no routing indirections in place.

Examples of services that perform large scale data mining for this type of information include the Mozilla Location Service and Google. The former provides celular information available for free[1], however wifi access point information is not provided in bulk, due to privacy and legal concerns.

These services gather information such as IP addresses in open networks, wireless or celular access point identifiers, and even short range communication identifiers such as bluetooth. This information is used in many benign services for location information

[1]https://location.services.mozilla.com/downloads

without the use of GPS, a feature commonly seen in mobile devices, for location based services.

Generally speaking, for privacy purposes, the ability to hold multiple identifiers at the same time and to assign new ones at will translates into increased privacy control. Abstractly, any mechanisms that conceal network identifiers can be used for this purpose, such as Virtual Private Network (VPN) providers or proxies. These require some type of network support, through intermediate network nodes, or general purpose privacy networks such as The Onion Router (ToR). Another option is to use different addresses over time. This is sometimes referred as pseudonymity, when a network device will hold multiple addresses, as if multiple devices were present. However not all networks allow this type of approach, often restricting the user device to a single IP address per device. MAC address rotation is now common practice in mobile devices, to avoid tracking of unconnected devices, and while it is not fully effective [15] against timing analysis, it is a welcome strategy.

1.2.2 Privacy in the face of novel network semantics

Novel network architectures based on ICN support the use of different semantics at the network layer. One of the primary drivers for this type of approach is to enable network caching of content to conserve bandwidth.

Some propose the use of content derived names (such as using content hashes) as network routing addresses, increasing the number of addressable nodes in the network [23]. The immediate benefit of this approach is to enable self verifiable names, since content authenticity can be verified by hashing and comparing it with its name.

Another approach consists on using hierarchical identifiers as network addresses, much like URL paths. Furthermore, since arbitrary amounts of data can be used, content hashes can also be included within the name and used for verification. This holds true not only for standard network operations but also for constrained IoT environments where standardisation efforts use Uniform Resource Identifiers (URIs) (or part thereof) for identification [24, 25] or location [26]. Finally ICN frameworks like CCN[6] and Named Data Networking (NDN)[7] combine both approaches, embedding hashes in hierarchical names.

These novel approaches contrast with traditional network layer addressing, that tries to use short fixed length network addresses. As such, some of the mitigation mechanisms described earlier do not apply to these protocols. Pseudonymity mechanisms would have to deal with a much larger amount of identifiers, and keeping large state mapping tables is not viable. Furthermore this type of addressing has different privacy implications. While an IP address discloses the intended communication endpoint location. Content addressing may disclose what content is exchanged. The closest

example to this type of semantics usually happens at the application layer URLs. But these convey human semantics, such as data description, purpose and state. This is not a property of URLs, but rather the result of established practice, for human purposes.

There is a considerable amount of mechanisms at the application layer to prevent privacy disclosure, through end-to-end encryption, and client side state scrubbing. However those are not directly applicable to network protocols, because network addresses are clearly visible to all intermediate parties, a feature that ICN leverages for in path caching. By definition, ICN exposes additional information (content) to a wider scope and furthermore causes a shift in responsibility. The content consumer does not get to choose how content is named, so it can not enforce privacy policies other than refusing to retrieve content or resorting to network supported concealment mechanisms such as a VPN. In ICN these could be name anonymizer services [27] or at least name scrubbing techniques that produce names without metadata [28].

1.3 Hypothesis and Objectives

From the previous observations two aspects should now be clear. First a large number of namespaces and associated resolution protocols are currently in use throughout different networks and environments. Second, naming plays an important role in composing computer networks and services, often defining how disjoint parts of the network are interconnected, with privacy implications beyond the network layer.

Based on empirical evidence it is expectable that network convergence will continue. Although novel protocols and/or architectures may eventually shift the status quo, since the Internet is the support infrastructure for the global economy, abrupt changes are not the expectable. Economic incentives dictate that resources will eventually be connected to the Internet, no matter how. The questions that must be answered then, is how to assign names based on these realities, how to resolve them, what are the consequences of our choices in the binding of names, and finally, whether this is achievable over existing protocols.

Assuming this is possible, one is left to ascertain to what extent, at what cost, and by what means. This work can then be structured around the following objectives, in the context of a continuous migration trend towards novel networks:

1. Develop interoperability mechanisms that facilitate network namespace integration across different architectures.
2. Apply namespace assignment strategies that improve name privacy at the network layer and beyond, in and across different architectures.
3. Apply content naming semantics to existing discovery protocols to bootstrap security mechanisms in legacy environments.

1.4 Contributions

In this section, a short overview of the outcomes of this PhD is presented. Much of my research prior to starting this PhD assumed privacy at the network layer was a sustainable mechanism [29], with the upper layers handling these issues using protocols such as Transport Layer Security (TLS) [30] for confidentiality or anonymity networks such as ToR for anonymity. My initial PhD work started within the 7th Framework Program (FP7) project Secure Widespread Identity for Federated Telecommunications (SWIFT), with the goal of introducing resolution mechanisms over its architecture [31, 32, 33] for interoperability with existing architectures, while exploring privacy mechanisms at the network layer [34]. These two topics are closely related to this PhD, and are the key lines over which one can outline the contributions done throughout this work.

SWIFT operated under the assumption that resolution systems need to function over pre-existing security and trust infrastructure [35]. Practice shows that outside of heavily controlled environments this assumption does not hold, either because the original protocols that support the Internet were not designed with security and privacy in mind, or because services will not deploy dedicated security infrastructure.

This led to a shift from the centralised scenarios in SWIFT onto more decentralised environments, where assumptions about the network tend to break. If global connectivity is not guaranteed, and centralised resolution systems are not available, communication still requires mechanisms to discover network entities based on trust relations that are transparent to the protocols. These problems might appear to be corner cases, but they are common in mobile environments where the norm is device to device communication. This problem was initially tackled under Portugal Telecom Inovação funding with the goal of introducing mutual authentication in mobile scenarios such as machine-to-machine security and e-Ticketing [36]. Further technical details were covered in [37], including Bluetooth and IP protocols, as well as physical discovery of digital services based on signed QR codes to bootstrap secure transport protocols.

While this work established methods to bind high level Identity Management (IdM) mechanisms with machine to machine communication as seen in mobile environments, it still lacked the generality seen in shims, such as HIP. This work was then extended, in [38], and approaches the general problem of secure naming in cross protocol discovery. Without changing the underlying protocols it imbues data in the underlying discovery namespace, in order to provide additional features and security semantics. Finally this approach was also extended to support the NDN protocol. Since NDN (and CCN) lack, in general, truly decentralised discovery protocols, the initial work included the definition of generic mechanisms for discovery in NDN [39] followed by the extensions

of previously designed interoperability mechanisms to support this protocol[40, 41].

The problem of privacy leakage as part of network protocols can be understood through the study of the relations between user information and its relation to network concepts [42]. However the role of non explicit relations is hard to analyse without deep knowledge of the protocol in question. For a concrete example of this aspect, [43] carries out a study of common disclosure vectors, based on a mix of protocol vulnerability, human error and accidental leakage of names across distinct protocols and how these end up revealing user location.

As a starting point for privacy integration studies, the Domain Name System (DNS) was targeted as a subject of study, with the introduction of generic mechanisms over DNS even if sacrificing some of its legacy benefits [44]. This enabled the integration of other authorisation mechanisms with DNS, which could in turn be used to enforce privacy controls. But, in general, the specific case of DNS privacy has been sufficiently addressed in both practical solutions [45, 46] and through standardisation efforts [47]. Even if these solutions are not widely adopted they do offer a clear path for migration.

When going up the network stack to the application layer, one finds that hierarchical namespaces are increasingly adopted, with DNS and URLs being the classical examples. Conversely this type of naming is also adopted at the network layer by emerging network architectures such as CCN and NDN. This further motivates the need for privacy mechanisms on top of hierarchical namespaces, a work carried out in [48]. Its main implementation is focused on the Hypertext Transfer Protocol (HTTP), meaning that it is implemented for DNS hostnames and URLs metadata. This choice was made primarily for practical reasons, since there is an abundance of existing on privacy leakage under these namespacest, and these are straightforward to demonstrate over existing technologies. However it should be pointed out that these are equally applicable to ICN architectures such as NDN and CCN.

For reading convenience the full list of publications carried out throughout this PhD is now summarised:

[34] Alfredo Matos, Rui Ferreira, Susana Sargento, and Rui Aguiar. "Virtual Network Stacks: From Theory to Practice". In: *Wiley Security and Communication Networks* 5.7 (July 2012), pp. 738–751. DOI: 10.1002/sec.368.

[35] Rodolphe Marques, Rui Ferreira, and Alfredo Matos. "Cross Layer Privacy Support for Identity Management". In: *Future Network and Mobile Summit*. MS10. Florence, Italy, June 2010.

[36] Rui Ferreira, Alfredo Matos, Goncalo Morais, Rui L. Aguiar, Pedro Santos, and Ricardo Pereira Azevedo. "Multipass: Gestão de e-Tickets em Dispositivos Móveis". In: *Revista Saber & Fazer Telecomunicações* 9 (Dec. 2011), pp. 76–81.

[37] Rui Ferreira, Alfredo Matos, Susana Sargento, and Rui L. Aguiar. “Multipass: Autenticação Mútua em Cenários Heterogéneos”. In: *Proc Conf. sobre Redes de Computadores - CRC*. Aveiro, Portugal, Nov. 2012.

[38] Rui Ferreira, Alfredo Matos, and Rui Aguiar. “Recognizing Entities Across Protocols with Unified UUID Discovery and Asymmetric Keys”. In: *IEEE GLOBECOM*. 2013.

[39] José Quevedo, Rui Ferreira, Carlos Guimarães, Rui L. Aguiar, and Daniel Corujo. “Internet of Things discovery in interoperable Information Centric and IP networks”. In: *Internet Technology Letters* 1.1 (2018). e1 ITL-17-0001.R1, e1–n/a. DOI: 10.1002/itl2.1.

[40] José Quevedo, Carlos Guimarães, Rui Ferreira, Daniel Corujo, and Rui L. Aguiar. “ICN as Network Infrastructure for Multi-Sensory Devices: Local Domain Service Discovery for ICN-based IoT Environments”. In: *Wireless Personal Communications* 95.1 (July 2017), pp. 7–26. DOI: 10.1007/s11277-017-4425-7.

[41] Daniel Corujo, Carlos Guimarães, José Quevedo, Rui Ferreira, and Rui L. Aguiar. “Information Centric Exchange Mechanisms for IoT Interoperable Deployment”. In: *User-Centric and Information-Centric Networking and Services: Access Networks and Emerging Trends*. Ed. by M.B. Krishna. Taylor & Francis Group, 2018. Chap. 3.

[43] Rui Ferreira and Rui Aguiar. “Breaching location privacy in XMPP based messaging”. In: *IEEE GLOBECOM*. 2012.

[44] Rui Ferreira, Alfredo Matos, and Rui Aguiar. “Hint-driven DNS resolution”. In: *IEEE symposium on Computers and Communications*. ISCC’11. Corfu, Greece, 2011.

[48] Rui Ferreira and Rui L. Aguiar. “Repositioning privacy concerns: Web servers controlling URL metadata”. In: *Journal of Information Security and Applications* 46 (2019), pp. 121–137. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2019.03.010.

1.5 Structure

The main work of this Thesis is organised around two axis in line with the objectives defined earlier: the construction of namespaces for interconnecting computer networks, and privacy mechanisms for network identifiers. These are treated as independent topics, for the most part, but the unavoidable relation between them is often discussed throughout the document.

This thesis is then structured around 6 chapters, and each intermediate chapter starts with a brief context introduction and ends with a summary or conclusions. Chapters 2 and 3 contextualise the topics and identify related work, with Chapter 2 outlining the main concepts and terminology to be used throughout the document, and Chapter 3 providing an overview of the various namespaces and resolution architectures available.

Chapters 4 and 5 discuss the main contributions of the thesis, structured around the two topics enumerated earlier, respectively the construction of secure naming from discovery protocols and privacy mechanisms introduced as name assignment policies.

Chapter 4 deals with the embedding of security semantics in existing discovery protocols. The focus is on overlaying additional security information over existing namespaces used by traditional discovery protocols and deriving other benefits that go beyond security features and can be applied in the upper layers.

In Chapter 5 a group of privacy concealing mechanisms is proposed, starting at the network layer and going up to the concealment of private information in URLs. For practical implementation purposes these solutions are instantiated over common protocols, but the discussed approaches are equally applicable in other contexts.

As the closing chapter, Chapter 6 summarises the achievements of this thesis, highlighting the main insights from this work, discussing them within the context of network privacy and network interoperability, while proposing directions for future work in related areas.

Chapter 2

Naming in Computer Networks

There are only two hard things in Computer Science: cache invalidation and naming things

Phil Karlton

This chapter outlines the core concepts about naming that are the subject of study, defines a terminology and identifies present uses of name resolution that are relevant in the scope of this work. A more detailed technical description is made in a later chapter, while the goal here is to pin some notions into place and provide context.

2.1 Introduction

Computer networks can be seen as a graph of connected resources, where naming, much like in human language, works as a tool for resource sharing. The primary function of a network is to move data across individual nodes, and this would be much harder to achieve without the means to express source or destination.

Historically names in computer networks have a strong relation to the topology of the network, as they did in old telephone systems where line numbers were used in phone numbers (before the advent of the digital network). This interdependency results from the need for efficient routing in the network being dependent on route aggregation, a limitation known as "Rekhter's Law" [49, 50]

> Addressing can follow topology or topology can follow addressing. Choose one.
>
> – Yakov Rekhter

Even in today's networks this remains in effect, issues such as mobility are handled through indirection mechanisms, either explicitly such as Mobile IP (MIP)[51], or implicitly such as Proxy Mobile IP (PMIP)[52] when the node itself is unaware of the topology change. Regardless of the process, a node should not hold an address that does match its current routing location, without some kind of indirection. Given this topological notion of locality, the names of the nodes in the network are often called addresses.

Before going into more detail, it is necessary to further describe what types of objects need to be named in a computer network. A classic reference in the field arrives from John Shoch [53] that categorises identifiers according to their purpose inside the network into three distinct types.

> The name of a resource indicates **what** we seek, an address indicates **where** it is, and a route tells us **how** to get there.

This very pragmatical description fits well into a layered view of the network (Figure 2.1). We often reason about computer networks and protocols in layers of functionality with well defined interfaces binding them together. From that perspective its easy to assume addresses are always *Network Layer* attachment points, such as IP addresses. And names under this definition belong to the *Application Layer*.

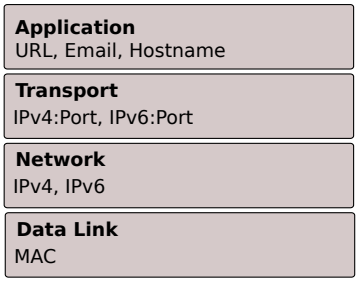


Figure 2.1: Different types of identifiers across the TCP/IP stack

However Saltzer [54] points out that names can appear at different levels of abstraction or under different representations, while reasoning about names without context leads to little insight, and concludes one should not reason about names in isolation but in the context of what object they bind to. In particular Saltzer[54] considers the definition from *Shoch* [53] leaves room for interpretation, as notions of what/how/where depend on external definitions of resources and the networking functions, that may or may not overlap.

Practice shows that nothing prevents the use of an address as a name, or a name as an address or as route, provided the necessary operational conditions hold - in fact this often simplifies the inner working of different layers. For example IPv6 addresses can be defined based on unique link layer identifiers [18], but this makes the device

globally identifiable [19]. Similarly the URL syntax [55] allows both DNS hostnames and IP addresses as part of the host segment, otherwise it would be impossible to refer to a resource without an existing DNS binding. This concept is straightforward from a natural language perspective and pragmatical from a network design perspective - i.e. *we point at things we cannot name*. When lacking a registered name for a host, rely on the network's forwarding functions. The consequence of this approach is that identifiers often *bleed out* of their intended design layer. As such, [56] insists on names being considered as opaque outside their own namespace, but this is often not the case as concepts from different layers get pushed up or down the stack. The Host Identity Protocol (HIP) [10] is one such case, where IPv6 addresses are generated from cryptographic keys, but more generally the concept can be applied to other types of data [57].

This chapter takes us through what name resolution is (Section 2.2), the characteristics of names in computer networks (Section 2.3) and finally how this relates to network functions (Section 2.4 and Section 2.5).

2.2 Name Resolution

Before defining what name resolution concepts are relevant for this work, some common terminology is required. To avoid confusion this text will follow the definitions from [58] loosely transcribed as follows:

- A *Namespace* is a set of names from which all names for a given collection of objects are taken.
- A *Name* is a unique string, in some alphabet, that unambiguously denotes some object.
- The *Scope* of the namespace is a function which defines the class of objects that can be named with elements from that namespace.
- The operation *Assignment* allocates a name in the namespace for later use.
- The operation *Binding* binds a name to an object. More than one name may be bound to an object.
- A *resolution* function defines the mapping of elements in the namespace, to the objects in the scope.

Name Resolution is the process used to determine the object a name binds to. In a computer network this is a distributed process that may span multiple systems until it is completed. This is sometimes referred as looking up a name, because generally there are two strategies for resolving a name into the corresponding object [58]:

1. exhaustive search across all objects (or discovery)

2. the name holds information that narrows the search

Notice that [56] takes this a step further when referring to computer networks, and states that 'Name resolution is always neighbour discovery'. This is not the case for all name resolution systems, but it seems to be the case for systems where names are strictly network addresses because resolving a name implies reaching the correspondent network element. What a name addresses defines the resolution function that is used. When resolving IP addresses, the network routing functions can be used to obtain more information about the owner of the IP. If routing is not available, less efficient methods, such as broadcast or multicast messages are used. In other contexts, such as databases or filesystems, the namespace might be the same but the resolution function or the scope are different.

Most complex name resolution systems support some form of indirection where a name resolves to another name through an intermediate object, but the end result is still an object. This facilitates the management of the namespace, through recursion relationships between objects, but sometimes obscures the real relation between the namespace and the scope.

A quick remark about language: depending on the reader background the terms 'name' and 'address' may hold different meanings. In some contexts 'name' refers to a human memorable representation, while 'address' refers to a binary representation of the same information. No such distinction is made in this text, 'name' is used in the broader sense of the word with no distinction.

2.3 Namespace Characteristics

The previous definitions have covered the functions and characteristics of name resolution systems, but have not covered the characteristics of the names in the namespace. When concerning the assignment, binding and resolution of names a number of characteristics of the names in the namespace are involved in these processes:

- **Hierarchical vs Flat** A flat namespace holds no structure with regards to the resolution, assignment or binding of the name, all these processes are identical for all names in the namespace. Hierarchical namespaces have internal structure that affects these processes. Hierarchy within a namespace is often used during resolution to delegate parts of the resolution process to different entities (e.g. for scalability). It can also be part of the binding and assignment of names, to delegate responsibility for resolution of the namespace. For example in DNS assignment of a name will differ under each Top Level Domain (TLD), with distinct rules and limitations.

- **Uniqueness:** Is a name bound to a single object? Or can a binding change over time. Uniqueness depends on the binding function to define which objects are bound to a name and how the binding can be changed. For example in a namespace where names are hashes [59], the binding function restricts the use of the name to a specific object (or set of objects) that match that hash, and no other binding can occur.
- **Fixed or Variable length:** Is the length of a name bound by a limit, or is there no limit to the size of the namespace? In some network protocols (such as IP) names are limited in size to conform with protocol message size limitations or resource limitations. In other namespaces (like URLs) the length is highly variable. The later case means that potentially the namespace could have an infinite amount of names, but in practice it is limited by protocol message sizes, or other constraints.

A conjecture put forth by Zooko Wilcox-O'Hearn (and often called Zooko's triangle [60]) is that names in a namespace can only have two out of the following three desirable characteristics:

- **Decentralised:** No need for a centralised trust authority to operate the allocation, binding and resolution of names.
- **Secure:** An attacker cannot subvert the resolution process to return an incorrect value.
- **Human-meaningful:** some namespaces are designed for human use, in the sense that names can be assigned based on user criteria, and are easy to memorise and compare because they convey meaning.

It can be seen that no traditional namespace currently holds all these three characteristics at the same time, and any system falls into one of three combinations:

1. A namespace that is both secure and human meaningful requires trust in a centralised authority (or multiple authorities under the same root). The fact that names are meaningful means they are subject to market notions of value, scarcity, speculation and trademark arbitration.
2. In a namespace that is decentralised and human meaningful, but not secure, any entity can claim any name it wants. This is the type of system seen in very basic computer networks, where any node can use any IP or MAC address with no enforcement. Naming collisions can occur.
3. In a namespace of decentralised and secure names, the name must be self authenticating (e.g. a hash of a public key used to sign bindings) such as [59], but these are not human-meaningful.

Some systems forgo human-meaningful names and settle for human-memorable names, that are short enough to be memorable, but the assignment of meaningful names is not provided. For example, proquints [61] represent any name as a spellable expression, and petnames [62] use a similar approach with dictionary words. But these forms of representation do not alter the assignment and binding operations, and despite being useful they are not applicable across cultures or languages. Other approaches rely on alternative notions of trust, or decentralisation models like the distributed ledger technologies [63]. While the later appears to achieve all characteristics from Zooko's triangle, this is still a matter of contention, as the notion of blockhain decentralisation might not match cleanly with Zooko's model [64, 65], and consensus problems have been observed in Namecoin [66] that support this limitation.

The ITU-T recommendations for future network identifiers [67, 68] defines the following set of characteristics as being desirable in future networks:

1. scope should be embedded in the name
2. the object category may be embedded in the name
3. a name must be unique under a given scope
4. the resolution function must be accompanied by security functions
5. names can be persistent or temporary

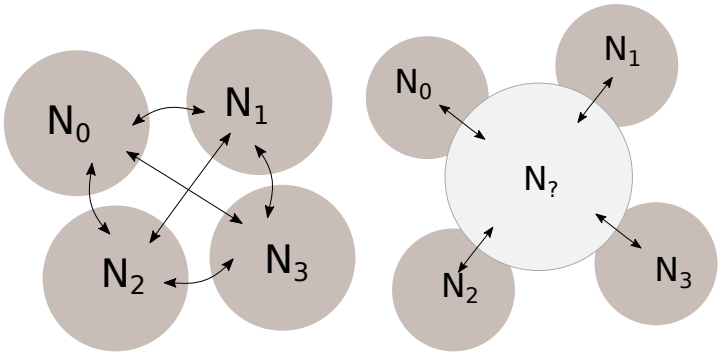
The previous list of characteristics also offers a good framework to reason about the limitations of existing identifiers, which ones lack these characteristics and how this affects their use going forward. However this list is orthogonal to Zooko's model. While the first two characteristics relate to the practice of embedding resolution information in names, the remaining three are not characteristics of any specific namespace but rather characteristics of specific binding or assignment functions.

2.4 Network expansion as namespace composition

Sharing of resources in a network works through the routing of data across addressable (i.e. named) nodes. Previously, network addresses were used to identify individual computer links, but this view is not entirely accurate since some computer could have multiple links (i.e. multihoming). Furthermore, network virtualization mixes real computer networks with digital constructs that emulate them. As such, a network is a logical resource which may not map directly to the physical hardware organisation.

As the network grows, one needs to add more names for the new nodes, or rebind unused ones. Generally, any novel network protocol that targets interoperability with existing protocols will eventually attempt to interoperate with existing networks. Ideally all names from a network would be available in another (Figure 2.2a), but this is

not always possible and some intermediate namespace may be used as an intermediate resolution namespace (Figure 2.2b).



(a) Fully connected (b) A super namespace

Figure 2.2: Namespace composition strategies

This problem can be reduced to the unidirectional connection of two different networks, i.e. both scopes are reachable from a single namespace. The techniques that can be applied to achieve this depend on the characteristics of the namespace in particular on the size of the namespaces being connected.

2.4.1 Variable Length names

Looking back at early telephone networks, different areas operated independently before they became interconnected, as did different countries before international country codes were assigned [69]. One of the challenges faced then was how to interconnect different networks, and defining rules to avoid ambiguity when routing calls. To solve this problem a new namespace was created (Figure 2.3) from the local namespaces, assigning each a different prefix code, while keeping the remaining number unchanged.

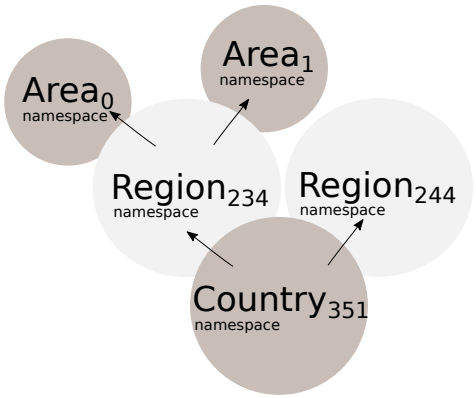


Figure 2.3: Combining multiple namespaces through a new namespace

There are two characteristics of the telephone number namespace that enable this solution: first, phone numbers are variable length names; and second, the hierarchical

organisation of the phone network matches the hierarchical assignment of numbers (phone numbers are routes across the network). Provided these assumptions hold, this same approach can be used recursively, and historically this is what happened as multiple areas and later multiple countries became interconnected. The previous approach also provides a benefit for users, in that they get to "keep" the same phone numbers, because the number assigned under the new namespace is based on the previous number.

This facilitates resolution of a number into its local counterpart, because it is already part of the full number. More generally, for each name in one namespace there is one corresponding name in another namespace thus transitive resolution from the first namespace to the scope of the second is possible. However the reverse is not possible without additional context: resolving a local area phone number into a full number that can be used in international calls requires knowing the relevant country code.

Because computer networks are not static constructs, they grow with time as more nodes become available. In telephone networks, the naming of the nodes followed the topology of the network, and growing the network was done through the addition of nodes that were assigned a new prefix. Creating this composite namespace was made easier because the names from the original namespaces where admissible as part of the names in the new namespace. Composition of hierarchical namespaces is well understood and is a common form of organisation not only in computer networks, but also in file systems [70]. URLs are composed from DNS names, filesystem paths and protocol names; UNIX filesystem paths can be routes in a local filesystem or onto remote machines; and IPFS [71] composes paths from other namespaces.

Assuming an infinite length, such names can include names from any other namespace. In practice size limitations constrict this ability, either due to protocol implementation or resource availability and as such one needs to consider interoperability with fixed length namespaces. However, by including the first name within the second, this approach reveals information which was previously only available under a more reduced scope i.e. any private information that was part of the name bleeds out of its intended scope.

2.4.2 Fixed Length names

For historical and performance reasons, many of the addresses used in the lower layers of the network are flat fixed length names. In the IP network stack all network identifiers from the link to the transport layer are designed this way (MAC addresses, IP addresses and TCP/UDP ports). At the application layer, namespaces such as DNS hostnames and URLs are variable length hierarchical names. Other types of flat fixed length identifiers can also be seen at the upper layers due to their unique binding char-

acteristics. For example UUIDs and cryptographic hashes are bound to certain objects, and this makes them ideal for some applications.

Thus composition of namespaces with fixed length names follows different strategies. With two namespaces of different sizes, the smaller namespace cannot possibly define a resolution function for each name in the larger namespace at the same time. The solution to this problem is often to have the binding between namespaces to be partial, temporary or based on context. For example, during the specification of the IPv6 protocol, a number of mechanisms were created to allow interoperability with IPv4 networks, that demonstrate this constraint. Because IPv6 addresses are larger than IPv4 addresses, one of the solutions is to include the full IPv4 address as part of the IPv6 address using a well known prefix for IPv4 hosts [72, 73]. This facilitates the implementation of interoperability gateways used to connect both networks, because the binding is stored as part of the IPv6 address. The reverse process is not possible due to size constraints, as an IPv6 address cannot fit in an IPv4 address. This requires gateways to retain state in the form of bindings between the names in IPv4 to names in IPv6, and handle these translation [74, 75]. Since this type of approach is known to cause problems [76] with application layer protocols that use addresses directly in their payloads, other solutions based on protocol encapsulation [77] are used to avoid them (i.e. through the use of routing overlays).

2.5 Name resolution from network routing

This chapter started with "Rekhter's Law", stating that topology must follow addressing or vice-versa. Routing in computer networks can be used to trivially implement name resolution for the namespace of node names, because one can query the owner of a name by sending a message and waiting for a response from the node with the resolved object. In this way, routing functions are used as an implicit mechanism to verify ownership of a name and perform name resolution. When routing is not available, discovery mechanisms can be used instead. In other words advertisement fulfil the role of the binding operation, and message forwarding works as part of the resolution mechanisms.

However, what happens when the namespace provided by the network does not hold the necessary characteristics? The namespace is not human memorable, or not large enough, or the network topology does not correspond to the intended meaning of the desired namespace.

The solution to this problem is to build a new network, with a new topology that matches the intended features. But throwing away the old network may also discard its infrastructure and desirable features, which is likely impractical from an economic

point of view. A more realistic approach is to build an overlay network, a group of nodes inside the network that establish logical links between them and route messages using a new namespace. Since the links in the overlay network are logical links, its topology is also logical and may or may not be similar to the topology of the underlying network (Figure 2.4).

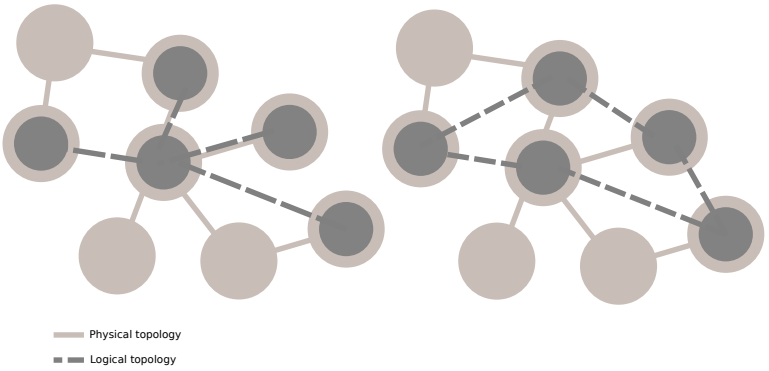


Figure 2.4: Different overlay topologies over the same network

Nodes is this new network need to agree on a common protocol for advertising availability and propagate routing information. For small networks networks, discovery protocols can be used for this purpose. In larger networks, nodes can use bootstrap nodes (well known nodes with high redundancy) to attach into the network.

When the underlying network lacks the desired characteristics a specialised overlay network for name resolution can be constructed to resolve a new namespace into network locators. This was what happened with IP network since IP addresses lacked human meaningful names, DNS emerged as a hierarchical network of servers that manage a centralised human meaningful namespace.

The recent trend of ICN networks introduces, again, a new resolution namespace. In the past, Distributed Hash Tables (DHTs) were often used for the construction of name resolution overlay networks for ICN names. Novel network architectures such as NDN, Data Oriented Network Architecture (DONA), eXpressive Internet Architecture (XIA) or Publish Subscribe Internet Technology (PURSUIT) introduce general purpose packet switching architectures that also introduce new types of namespaces. For now it is still unclear if routing in these architectures is sufficient to provide all the necessary features expected of a namespace, and one can find proposals in the literature for both routing independent name resolution protocols as well the inclusion of name resolution mechanisms as part of routing signalling for these architectures.

2.6 Summary

This chapter covered basic terminology and high level concepts on how namespaces are defined from the network or, conversely, how the network can be defined from a namespace.

Resolution is the process of determining which object a name binds to. In a distributed computer network this involves matching the name being resolved with a mixture of discovery information that defines the network topology.

Not all network addressing namespaces are identical. Some use variable length names, which could (in theory) hold an infinite amount of names, others are limited in size which means assignment of names is temporary and global uniqueness is not possible. Furthermore, the information placed in names influences the utility of the namespace, whether because it assists in the resolution process or because it provides external value such as human usability or security functions.

At this point it should be clear that network composition and naming are closely related. Because name resolution can be partially modeled as network routing, then network overlays can be used to implement name resolution systems that use a different namespace from the underlying network. Likewise, extending the network with the addition of new nodes can alter the resolution of names in the network, and can be seen as a particular case of composition.

Variable length hierarchical namespaces can be used for composition of multiple namespaces, and in fact express routes across a network. Fixed length namespaces need to resort to other approaches, such as restricting uniqueness in time or based on other implicit context from other sources. Regardless of the network, the fundamental limitations for resource exhaustion and extension still apply. Recursion can conceal the complexity of the network but it does not eliminate it.

Based on these principles, the upcoming chapter tackles the more technical aspects of existing naming systems as the groundwork for the main contributions in the later chapters. Some are based on discovery protocols, others use network independent name resolution services and others are part of routing in ICN network architectures. Notwithstanding all should fall comfortably onto the characteristics discussed in this chapter.

Chapter 3

Name Resolution and Discovery Protocols

Anyone who considers protocol unimportant has never dealt with a cat

Robert A. Heinlein

Now that the core concepts were discussed, one can look into the existing state-of-the-art. The different protocols, namespaces how they fit into the earlier discussion and their relevance going forward.

3.1 Introduction

In the previous chapter, the main characteristics of a namespace and its resolution functions were identified. One can now look at the existing state of the art in protocols for discovery and name resolution architectures, and how they match these characteristics.

Name resolution protocols build upon the discovery or routing functions provided by the underlying network. The end result is always a namespace of names with a corresponding scope of resolvable objects. To begin, this chapter looks into general purpose protocols for name resolution, in particular those that establish their own namespace, rather than those that derive it from the underlying network protocols and what are their properties or limitations.

For clarity these protocols are divided into separate sections, based in which protocols they build on. Section 3.2 looks into what are commonly called discovery protocols, that work primarily through network broadcast and multicast functions. Section 3.3 covers full fledged name resolution protocols that are built as overlays over existing computer networks.

New namespaces, as introduced by alternative network architectures that borrow concepts from ICN are covered separately in Section 3.4. These architectures introduce new types of name assignment and binding, however it is still unclear which role they play for the future of networking. While they currently position themselves at the network layer they take advantage of data naming information to improve network efficiency while exposing this information to a much larger scope.

Finally, Section 3.5 discusses some of the implications of these namespaces, Section 3.6 summarises the contents of this chapter before moving forward into the main contributions of this thesis.

3.2 Discovery Protocols

For any network, discovery is the first step to determine the availability of other node. There are multiple protocols for this purpose [78]. Some protocols provide simple mechanisms to enumerate all nodes, others offer more complex mechanisms to retrieve information based on filters.

In small networks, nodes can rely on mechanisms such as broadcast or multicast messages to enumerate available nodes. For larger networks, other mechanisms can be used to simulate a broadcast medium, e.g. a central point of contact can be used to publish discovery information and handle discovery queries.

Naturally not all discovery protocols are meant for the same purpose and define distinct namespaces and scopes accordingly. Some are concerned with device enumeration, and to provide a simple list human memorable names and associated network locators. Others provide additional service metadata, such as a list of all services provided by a specific node in the network.

This section covers a cross layer selection of discovery protocols, it starts with general purpose broadcast/multicast advertisement protocols, and moves up the stack onto scopes that employ general purpose URLs and are used in web based services.

3.2.1 Bluetooth Service Discovery

Bluetooth [79] is a wireless protocol for communications over short distances. Each device is allocated a 48bit device address. On top of the wireless communication channel, it specifies a number of profiles and services for various purposes.

At the link layer, the Link Manager Protocol (LMP) provides two messages for devices to exchange human meaningful device names (up to 248 bytes). This enables a form of decentralised device naming where each device can claim any name.

For service discovery, the Service Discovery Protocol (SDP), enables devices to query their neighbours for associated attributes (service records). The protocol is synchronous and operates over the Logical link control and adaptation protocol (L2CAP).

Using this protocol, a client can query an SDP server for available attributes of different types. Universally Unique Identifier (UUID) attributes are used to characterise different services and clients can filter records based on UUID queries. Attributes can provide information about any service characteristic, most notability the channel identifier used to connect to the service.

Bluetooth Low Energy (BLE) (introduced with Bluetooth 4.0) defines a reduced stack for improved energy efficiency, that allows for broadcasting of data with no need for connection establishment. One of its supported features is the introduction of beacons, devices that wake periodically to broadcast small payloads (31 bytes).

Multiple competing protocols use this feature for device discovery under some namespace. iBeacon [80] advertises UUIDs that identify application types. The [81] protocol serves a similar purpose but can announce both UUIDs (in plaintext or encrypted) and URLs (up to 17 bytes).

3.2.2 Simple Service Discovery Protocol

The Simple Service Discovery Protocol (SSDP) [82] is a discovery protocol for IP networks. It relies on IP multicast to advertise presence and service availability. Service instances are identified by a Unique Service Name (USN) (i.e. a URI). A service description typically includes the USN (often built from a UUID), an opaque service type and a locator URL. Devices can query for specific service types using multicast queries that include the intended service type.

SSDP is the basis for service discovery in the Universal Plug and Play (UPnP) protocol stack, a common stack used in service discovery for consumer equipment that follow the interoperability guidelines from the Digital Living Network Alliance (DLNA).

SSDP lacks any intrinsic security mechanisms, and it is entirely dependent on security mechanisms implemented in other protocols (e.g. IPSec). Since the use case of the UPnP protocol stack is to support unstructured scenarios (home networks and small consumer equipment), in practice most devices do not implement additional security mechanisms.

3.2.3 DNS based Service Discovery

The DNS-based Service Discovery DNS-SD [83, 84] specifies a protocol primarily targeting local networks using IP multicast. This protocol reuses the namespace, record formats and messages used by DNS, but it defines a special TLD *.local* as referring to the local network.

This protocol is commonly used in local networks, such as home networks, by consumer equipments. In addition DNS-SD was initially implemented to work with IPv4 link local addresses [85], in networks that have no infrastructure to assign IP

addresses. This led to the adoption of this protocol in mobile environments as part of the Wifi Direct [86] protocol stack, a Wifi based protocol stack for device to device communication.

Clients can issue queries to look up the IP address of specific devices types (as *hostname.local*), or to query for services of a certain type. The later uses DNS service records in the form *_protocol._transport* to enumerate the different services available in a host (e.g. *_http._tcp* designates an HTTP service).

DNS-SD provides no security mechanisms other than those available in DNS (e.g. DNS Security (DNSSEC)). However since the protocol is commonly used in non structured scenarios such as device to device communication, with no previous trust establishment, one cannot rely on the availability of a Public Key Infrastructure (PKI).

3.2.4 Physical object discovery

Some discovery technologies are meant to associate digital information with physical objects. Two widespread examples of this are QR codes and Near Field Communication (NFC) tags.

QR codes are essentially bar codes that hold arbitrary information. These codes are particularly meant for recognition using cameras, being arranged in two dimensions with marker dots to facilitate recognition despite the orientation angle. The total capacity of a code [87] depends on the type of data and amount of error recovery, the maximum capacity is 4296 alphanumeric characters or 2953 bytes. In practice these codes are used to store various types of data: URLs are used to link printed media to web pages, bank information for money transfers [88], login credentials, or generic text messages. Intrinsically, QR codes do not support security mechanisms. Payload signatures or encryption is handled by the application that generates the bar code, e.g. [89] or other approaches as the one presented in Chapter 4.

For radio communication, NFC [90] enables low data rate communication between devices in close proximity (a few centimeters). This technology based on the Radio-frequency identification (RFID) Security in NFC, is available based on shared key encryption [91], On top of these interfaces, NFC provides three types of services: generic point-to-point communication with other NFC devices; smart card emulation, to interact with contactless smart cards e.g. for payments; and reading and writing of data storage tags that hold arbitrary records [90]. Much like in bar codes, the later capability is often used to store text and URLs, but NFC supports the inclusion of record signatures [92] to verify payload integrity based on included signatures and certificate chains.

For other communication mediums, similar mechanisms are also available, such as acoustic [93] or optical communication.

Since many of these solutions make use of passive equipment (printed barcodes, or passive NFC tags) that always produces the same payload, it is always possible for an attacker to engage in a replay attack, using another valid tag even if the payload is signed . For example, [94] takes advantage of this in vending machines to redirect users to the wrong endpoint, causing them to buy the correct products in a different vending machine.

3.2.5 HTTP based discovery

Given the popularity of the World Wide Web, there are several formats for discovery that leverage the URI namespace and operate on top of the HTTP protocol.

The Extensible Resource Descriptor Sequence (XRDS) is a format for describing resources associated to an URL. Typically this format is used to describe service endpoints associated with a URL, such as OAuth [95], OpenID [96] or Extensible Resource Identifier (XRI) [2]. Each service is characterised by a locator/identifier and a type which can be either a URI or XRI (Section 3.3.6). For security purposes XRDS documents may be signed with embedded XML signatures.

WebFinger [97] is a protocol for discovery of arbitrary information associated with a URI. A client queries a *well known* server for information about a resource URI, the server responds with a list of typed links. A link associates a locator to a piece of information of certain type [98]. This protocol is used as a service discovery mechanism for other protocols, for example to discovery a message inbox or identity provider from a user webpage or email address.

Both XRDS and WebFinger provide service discovery under a certain scope identified by a URI. This URI can be a URL for a webpage, an email or any other URI scheme. However this URI might not be sufficient to define how to reach a discovery server. Different applications employ a mixture of static rules and metadata embedded in other protocols. OpenID[96] uses metadata in Hypertext Markup Language (HTML) documents or HTTP headers to determine the location of a XRDS document. WebFinger constructs a well known URL from the hostname used in other URLs or email addresses.

3.3 Resolution Protocols

The previous section introduced several protocol and namespaces based on discovery mechanisms. However as the network grows, it can no longer rely on local network functions to perform name resolution. In order to scale, an efficient overlay for name resolution must be constructed.

Consequently, this section covers protocols that require some kind of overlay to perform name resolution. Following a similar structure to the previous section, this

starts with traditional architectures for network based name resolution and then moves on to DHTs and web based resolution systems that generically apply to URLs. This provides the needed context before moving on to the next section.

3.3.1 Domain Name System

The Domain Name System (DNS)[99, 100] provides a namespace of hierarchically organised, human-meaningful names. Initially it was intended as means to lookup IP addresses from names, but as the Internet grew it was expanded to support other types of objects.

Names in the DNS namespace are composed of segments, separated by dots, where each segment corresponds to administrative zone managed by a server that resolves into type records. Resolving a name yields a record of some type, the most popular ones being IP addresses, but it is also used to store small text records and public keys.

Administratively the namespace is managed as a tree of zones, where each zone is managed by a different entity The Internet Corporation for Assigned Names and Numbers (ICANN) oversees the registration of domain names in the internet, that is it controls the root of the tree and delegates the various zones to different entities. Each zone can delegate sections of its namespace to other entities that in turn can also delegate further. At the top of the tree there are general purpose TLDs such as *.com*, *.net* or *.org* but also TLDs assigned to countries such as *.pt*.

Names are read left to right, from the smallest zone up to the top of the tree, so *atnog.av.it.pt* is read as *atnog* under *.av* under *.it* under *.pt*, under the root zone (represented either as an empty string or a dot). The resolution process consists in going through all the segments from the right to the left, querying the responsible servers until it reaches the authoritative server, and queries for the intended type (Figure 3.1). Most IP access networks provide a local caching name server that will cache DNS records from previous queries, avoiding repeated queries to the various name servers, thus most terminals only need to query their local network caches to lookup a name.

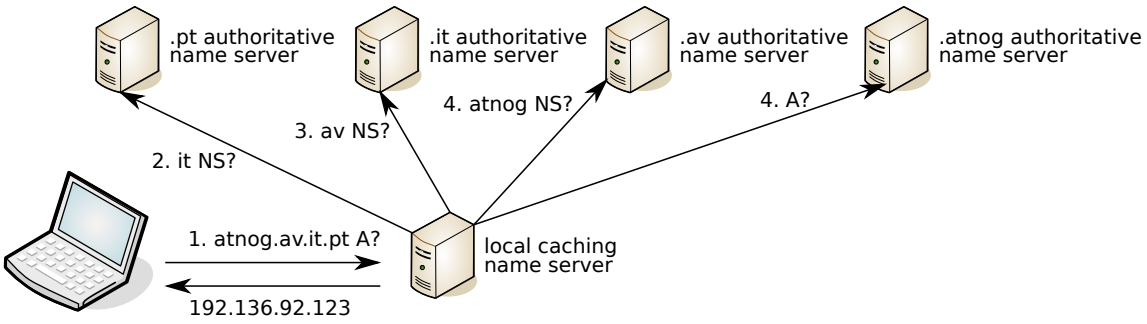


Figure 3.1: DNS resolution

In general DNS is considered to be a public database[101] in that it not does specify mechanisms for access control. Some implementations do provide access control mechanisms based on ancillary data (e.g. [102] based on the source IP address), however all data is sent in the clear using either User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) [103]. DNSSEC[104] was introduced to add signatures to DNS responses, avoiding forged messages or MITM attacks, however it does not address confidentiality.

DNS often clashes with privacy requirements, because its queries are sent in the clear [103]. DNS queries observed in the network disclose the hosts one interacts with, and possibly some additional information. The work in [105] demonstrates that aggressive DNS prefetching may even disclose user input, because it is erroneously interpreted as a hostname and immediately triggers a DNS requests of what the user is typing. The focus of the study conducted in [105] targets web browsers, but in fact a more general argument can be made for any software that may parse user input into network names. For example, [106] leaks all user input to the network, including passwords. As a consequence a number of privacy mechanisms to protect DNS information visibility have emerged, at different network elements.

To minimise this issue, [107] proposes that some DNS queries do not need to forward the full name in the query during iterative resolution, but should instead restrict it to the necessary labels. As an example, the root DNS servers do not need to know that a request refers to a specific web site, only that it refers to the *.com* TLD. This minimises information disclosure between the local resolver and upstream DNS servers.

DNSCrypt [46] is an encrypted transport protocol for DNS traffic, meant to conceal DNS information between the client and the resolver. While the protocol was never standardised, it is officially supported by several public DNS resolvers, including the OpenDNS[45]. More recently the Internet Engineering Task Force (IETF) drafted a similar alternative using TLS as transport protocol in [108], and HTTP Secure (HTTPS) [109] as a way to circumvent untrusted local caches. This has yet to pick up significant adoption by client or server implementations. Both these mechanisms increase privacy between the resolver and the DNS server, with regards to an eavesdropper.

3.3.2 Handle System

The Handle System [110, 111, 112] is a general purpose global name service. It defines a new namespace, as well as resolution protocol with security and management functions.

Names in this namespace are composed of two parts, separated by a slash (/). The first part, or prefix, is the naming authority and the second part, or suffix, is a unique name within the scope of the named authority. The naming authority consists

of multiple segments, separated by dots, where the leftmost segment is the top naming authority. For example *100.1002/atnog* is a valid handle, where the naming authority is *100.1002* and the local name is *atnog*. The naming authority prefix is itself a hierarchical namespace. In the previous example the naming authority *100* defines a child naming authority *1002*.

The architecture for the handle systems consists of a hierarchical tree of handle services. At the top level is the Global Handle Registry (GHR) that manages the handles assigned to other naming authorities or Local Handle Services (LHS). For bootstrapping, a client must always know how to reach the GHR, from which it can reach all other naming authorities. This information is usually included in implementations and signed updates can be retrieved from the GHR.

For security, the resolution protocol provides two way authentication between the client and the server as well as data integrity and confidentiality. Client authentication can use either passwords or public key cryptography.

In addition to the namespace, the Handle system also defines a data model that includes access control policies for each object associated to a handle, for fine grained access control. Objects associated to an handle are characterised by an index and a data type, and new data types can be added to support new types of data. A management protocol to manipulate this data model (e.g. update objects or access control policies), is also defined as part of the specification.

3.3.3 Distributed Hash Tables

Distributed Hash Tables (DHTs) are a type of distributed system where looking up of names resembles that of a local hash table which provides the ability to store and retrieve a value associated with a key.

Names are large integers with a pre defined size. Since names are flat, with no global notion of internal structure, the network topology follows from the namespace, and usually one can consider the name space to be circular (fig. 3.2). Each node divides the namespace into segments, based on the notion of distance to its own name (distance functions vary with implementations, some consider euclidean distance while others use XOR metric [14]). For each segment the node holds references (i.e. routes using the underlying transport protocol) to nodes within the segment that are used to route messages. Segments that are *closer* have more references than those more distant from the node.

Because the routing topology in the DHT may be unrelated to the network topology, and nodes are often at the edge of the network, routing is not optimal. The more general approach is for routing to require at most $O(log(n))$ nodes (where n is the number of nodes) [113, 114]. Other compromises can be achieved for better routing performance,

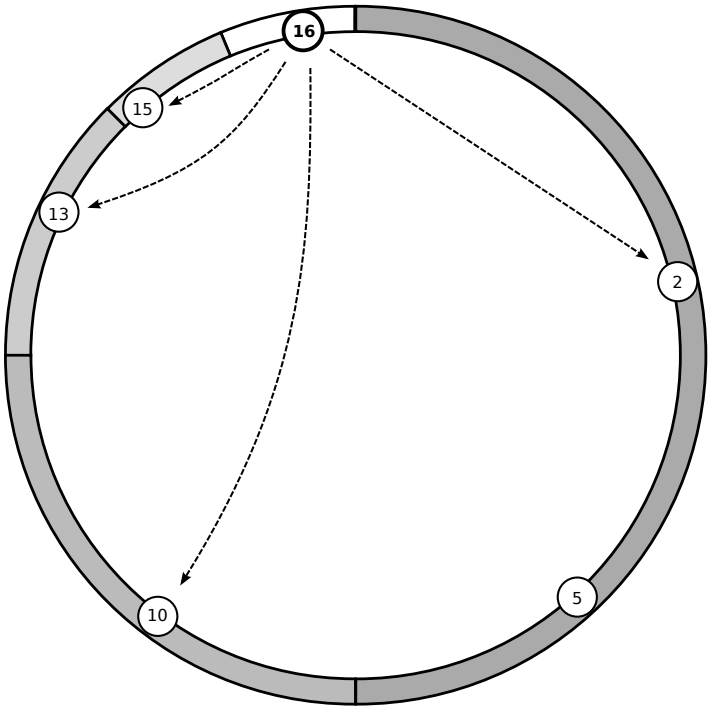


Figure 3.2: Topology of a DHT overlay network

either through the use of large routing tables [115], aggressive caching [116] or the introduction of an internal routing hierarchy within the DHT [113], and these can reduce the average time to amortised $O(1)$. Delay within the DHT, is hard to predict since routing may not follow the shortest path. Some implementations compensate this by weighting the choice of next hop of based on the previous round trip time [14].

Names in this type of namespace are not human memorable, but they are well suited for multiple applications:

- New nodes can choose their name using a random number generator (for long integers the collision probability is very low)
- Names can be the result of hashing a name in some other name space
- If the DHT is meant to store data, names can be hashes of the content

In addition, using a hash of public key as a name provides a trivial way to add security functions. The name can be used to retrieve the public key which is verified using the hash, and further communications can be encrypted using that key.

Multiple services make use of DHTs. In particular Peer to Peer (P2P) applications explore DHTs [117, 118], because they are decentralised and hold no single point of failure and generally all nodes are considered equal since any node can join the network.

For the purposes of network services, several solutions exist that are based on DHTs. The Internet Indirection Infrastructure (i3) [119] uses DHTs to provide traffic indirection services (e.g. anycast, multicast and application layer routing). Names

in i3 are hashes of DNS names, URLs or public keys depending on their purpose. Other proposals attempt to create an alternative to DNS using DHTs [120, 121, 3, 122]. However, DNS holds hundreds of millions [123] of top level domains, making it hard to support in a regular DHT without running into routing performance issues [124], requiring some stable nodes to expend bandwidth and storage to support active caching. Furthermore, the DHT namespace is not human-meaningful, although [122] avoids this issue by offloading the matching of human meaningful strings onto other tools (e.g. indexers or search engines)

It also stands to attention that despite their distributed nature, DHTs may not wield such resilience gains as expected, as pointed out by a study conducted in [124] DHT alternatives to DNS only outperform regular DNS when considering resilience to orchestrated attacks. If one assumes random failures them DNS still outperforms a DHT based solution. This comes from the fact that DNS has a very high level of replication for some of its nodes (closer to the root) in practise.

Some implementations combine existing namespaces with the namespace from a DHT. To replace the use of DNS hostnames in web URLs, [122] introduces Semantic Free Referencing (SFR) a new resolution system that resolves hashes of public keys into locators (IP, DNS, or another SFR hash). Its primary goal is to replace the use of DNS based URLs in web pages with URLs with a hash as in the authority e.g. $sfr://f01212099abcab678ac345ba4d.../path$.

Similarly to URLs the InterPlanetary File System (IPFS) [71] combines public key hashes with local file names, resembling a path of labels e.g.

```
/ipfs/XLF2ipQ4jD3UdeX5xp1KBgeHRhemUtaA8Vm/docs/ipfs
```

The *ipfs* prefix identifies the scheme and the second label (a hash of a public key) can be used to route requests in a DHT; the later labels are assigned by the holder of the key. To express content without going through the DHT, names are prefixed with IP addresses and TCP ports (or alternatively with DNS names) i.e. names are routes and each pair of path labels defines the scope and the name to be used next.

```
/ip4/104.131.131.82/tcp/4001/ipfs
    /QmaCpDMGvV2BGHeYERUEnRQAwe3N8SzbUtfsmvsqQLuvuJ
```

Since no DHT could be used to resolve it into a locator, the key hash used for verification is placed at the end of the path.

Likewise, the GNU Name System (GNS) [125] is a decentralised naming system, with two different namespaces: a local namespace (*.gnu*) is meant for user specific

names, only meaningful to the local user; a second namespace (*.zkey*) holds globally unique, cryptographically verifiable names (key hashes). A local process resolves between user names to global names and a DHT resolves global names.

```
alice.gnu
HEK9TEBUQ5AT5V3FLLOHHNDA12HGH6BIMO4TN7RDVOQ5B7TIEU8O.zkey
```

Names adopt the same format as DNS names, but some name segments are base32 hex encoded key fingerprints, used to verify if a zone is authentic. To enable interoperability with DNS, GNS provides DNS proxies that can be used with regular DNS clients. This is possible because GNS name and record formats are the same as used by DNS.

3.3.4 Security Assertion Markup Language

The Security Assertion Markup Language(SAML) is an OASIS standard for communication of user authentication and attributes. It is mostly used within the context of Identity Management, as means to provide Single Sign On.

SAML itself is not devoted to name resolution, but it must deal with entity identifiers in an appropriate manner and thus must provide some relevant semantics for name resolution. The SAML specification[126, 127, 128] provides three internal protocols that are relevant in the context of name resolution:

- Artifact Resolution Protocol
- Name Identifier Management
- Identifier Mapping Protocol

Being part of SAML these protocols can benefit from integrity, confidentiality and two-way authentication, properties inherently ensured by SAML.

The Artifact Resolution Protocol is provided in SAML as means to place references within messages, that can later be resolved into actual meaningful objects, mimicking the concept of "passing by reference". This is useful to prevent sending sensitive information over an insecure channel, postponing transmission of the actual content for a later time through a more secure channel. In SAML, references used in messages with this purpose are called artifacts and the Artifact Resolution Protocol handles the resolution of Artifacts into objects. Artifacts are by definition one-time-use identifiers.

The Name Identifier Mapping Protocol is used to renegotiate the use of an identifier between two entities. In the context of SAML, it is used for an Identity Provider to change the value of an user identifier used by one Service Provider or to deprecate the use of an identifier altogether. The protocol currently support two operations:

change the identifier associated with one entity, or terminate the use of the identifier completely.

The Identifier Mapping protocol is used to map identifiers between different namespaces. In the context of SAML, this allows two Service Providers to refer to the same user by mapping the different identifiers in the service providers that refer to the same user. The use of distinct namespaces for different Service Providers is a privacy protection feature, used to prevent user information disclosure.

3.3.5 Uniform Resource Locators

Uniform Resource Locator (URL) are a common locator format, most commonly associated with HTTP web pages. Their format is actually a composite from multiple other namespaces. As defined in RFC3986[55] an URL is composed as

$$Scheme : //Authority/Path?Query\#Fragment$$

Scheme: used by the client terminal to determine the protocol to be used.

Authority: a tuple *[User:Password@]Host[:Port]* that holds a server hostname, and optionally a port and user information. The *User* and *Password* are used for identification and authentication according to the service provider. The *Host* and *Port* are used to reach the server, and must refer to a valid network node (DNS hostname or IP).

Path: a set of segments separated by a slash. Segments "." and ".." denote relative references in the path.

Query: represents key/value pairs that further identify a resource.

Fragment: is used for client-side indirect referencing in resources associated with a URL. This means they are normally not seen as part of a request, and are used solely by the client implementation.

While RFC specifications do not impose strict limits on URL lengths, practical limits must be considered, based on common implementation practices, tools and protocols. A common denominator for an upper limit to URL length in various browsers and web indexing engines is 2000 bytes. This value is based on empirical observation of web browsers and HTTP server implementations, and while some implementations allow longer URLs (e.g. 100.000 bytes) this value remains a reasonable assumption for practical purposes[1]. An analysis of the Common Crawl[2] dataset that contains 1.603.205.557 unique URLs collected from web pages, only contained 299 URLs that exceeded 2000 bytes.

[1] So far the most up to date (2015) compilation of data on this subject can be found at http://www.boutell.com/newfaq/misc/urllength.html

[2] http://commoncrawl.org as of July 2015

Additional restrictions apply to the host name inside the *Authority* component. The total size for a DNS hostname must not exceed 253 bytes and in addition each label (between ".") is limited to 63 bytes. Furthermore domain names only accept a restricted set of characters and would require encoding schemes such as base32 [129] in order to represent arbitrary data.

Similarly the *Path*, *Query* and *Fragment* are also restricted in the the character set they can hold, each component has a different set of restricted characters. A one size fits all approach seen in several applications is the use of base64 [130], with a URL safe alphabet, to encode arbitrary data inside these components.

There are a couple of abstract specifications for name resolution over HTTP URLs. The World Wide Web Consortium (W3C) defines good practices for this type of systems[131], while the IETF specified the formats and functions [132] that such a system should provide. But no single standard resolution mechanism exists in practice and instead different services often implement HTTP applications that resemble name resolution services. Over the years, a number of online services have been created with the sole purpose of mapping single URLs into URLs managed by a different authority. These services are named "URL shortners", because their focus is to generate very short URLs for distribution over restricted medium such as SMS and Twitter. However there are similar services that focus on other properties, such as temporary URLs (both HTTP and email addresses) and on enforcing access control (e.g. link/address protectors that enforce password or captcha verification). Due to URL obfuscation, these are a viable mechanism for both privacy protection and a source of security issues [133]. Such services work well for distributing individual HTTP links but cannot be used for the general case, because redirecting an entire website domain would be the equivalent to an HTTP binding attack [134].

While URLs do not provide intrinsic security mechanisms, several applications embed information in URLs to perform security verifications. For example, [135] encodes signatures in URLs, and [136, 59] include hashes in URLs used to verify the resolution process.

3.3.6 Extensible Resource Identifiers

XRI is an OASIS standard for abstract digital identifiers syntax [2] and resolution. XDI.ORG, an international non-profit public trust organisation, provides a global registry for XRI identifiers enabling individuals or companies to register pairs of persistent and reassignable identifiers. Technically, XRIs are an extension of URIs, and share with them a similar syntax. XRI specifies the concept of persistent identifiers, that can never be reassigned. This is one of the main features that led to the adoption of XRI for user identification in OpenID [96, 137], because it provides both human-meaningful (but

temporary) identifiers as well as persistent identifiers that never expire.

XRI resolution is similar to DNS resolution, in the sense that both adopt a hierarchical architecture, although at different levels of abstraction. The XDI.org registry delegates the management of groups of identifiers to XRI brokers that in turn refer to other naming authorities. In addition XRI allow for nesting of identifiers, an XRI can include another XRI or an URL even at the authority level. This is used to express authorities which which are not registered (for example using an URL).

Since XRIs are designed as an extension of URIs, and the body of work regarding URIs is usually handled by the W3C, there has been some overlap of work between the two, in particular within the W3C TAG group in [131]. The current focus of W3C work seems to be on metadata retrieval rather than on name resolution.

XRI uses HTTP as transport protocol, enabling XRI to benefit from HTTP security standards like HTTPS or even to use Security Assertion Markup Language (SAML) [126] to provide confidentiality, integrity and authentication.

3.4 Alternative solutions

The paradigm shift brought by Information Centric Networks (ICN) is that network supported names are partially or completely independent of location and can be used to identify individual content in order to enable network caching. This shift is motivated by benefits to network infrastructure, such as lower latency, energy efficiency and mobility [138], features that benefit highly from network caching. Another side effect is that it becomes possible to verify content earlier as it is forwarded across the network, using naming schemes that include data hashes or provenance signature.

If the public World Wide Web is any indication, the number of existing webpages is over 3 billion web pages [3]. This challenge resulted in the emergence of multiple proposals for scalable Internet architectures, many which significantly modify the namespace of network names.

The focus of this section is the naming aspects of said architectures. A good survey on other aspects such caching, mobility and routing can be found in [23].

3.4.1 Data Oriented Network Architecture

In DONA, [4] names are made of two components *P:L*, a principal P and a label L, where P is a hash of a public key owned by the content publisher. The label L is arbitrary, provided they uniquely identify a piece of content; they can be hashes of the content for immutable data or some other name as defined by the publisher.

[3] According to the Common Crawl dataset from March 2018 `http://commoncrawl.org/2018/03/february-2018-crawl-archive-now-available/`

Each retrieved data includes the public key that generated P and is signed with the corresponding private key. As such, names can be said to be self-certifying, since these three pieces of information suffice to verify data integrity.

For name resolution, DONA relies on a hierarchical network of resolution handlers. Content publishers register the names they host with the closest handler, that then propagates this information up to the top tier handlers. The tier-1 resolution handlers have full knowledge of all registration in the network. Registration allows for wildcards to support aggregation of entries that point to the same handler. Resolving a name then requires going down the handler hierarchy until it can reach an handler that reaches the principal key holder.

The principal key is also used for security verification when registering new content, i.e. a registration for a name *P:L* must be signed by P, preventing the injection of fake entries in the resolution handlers.

Since in DONA names are not human memorable, users are expected to rely on third party search engine services to provide them with some type of mapping into human meaningful information.

3.4.2 Named Data Networking

The CCN [6] and NDN [7] architecture proposes a replacement for IP routing based on the Interest based routing, where a consumer sends an *Interest* message to a publisher (or intermediate cache) and receives a *Data* message routed via the reverse path. As such, data consumers of data do not need a routable address, and messages do not have source addresses, only a name and type that specifies their direction.

Names in NDN are hierarchical, often represented as paths or URLs (e.g. */atnog/av/it/pt*). However they are not required to be human readable, and any segment may include an arbitrary sequence of bytes. Individual name component have a type, either a free form sequence of bytes, a hash of the content, or typed numbers. The later type is meant for naming conventions already in use [139] in NDN, such as data segment numbers, content version numbers, timestamps and data sequencing.

Because routing is based on variable length names, name resolution can be implemented directly as routing of Interest and Data messages, with no need for dedicated name resolution infrastructure other than the implicit caching NDN already provides. However it is not clear if this approach is sustainable for two reasons: first NDN routers would need to hold routes for a very large number of routing prefixes, making global scale routing unsustainable; second a resolution overlay from the network enables mobility without the need to advertise routes (i.e. mobile clients without network supported mobility) and it is not clear if NDN names are truly independent of topology.

Despite the technical achievements of NDN, it is still unclear how top level prefix assignment would take place. In principle top level prefixes could mimic human organisations as seen in DNS top level domains, but this would mean hundreds of millions[123] of prefixes. As the debate over this topic progresses, one can already find proposals such as [140] that supports the name prefix in CCN should identify the Autonomous System (AS); this would reduce the number of prefixes to 60.000 [141], but would mean the name prefix is topology dependent.

Similarly, [142] supports that efficient routing cannot happen in CCN, due to long variable length name segments. It proposes such names should be transformed at the edge of the network into more efficient routing names, a transformation which is then reversed at the destination using reverse mapping functions. In this case, there is no clear separation of namespaces from the point of view of the consumer, as the access network handles the transformation of names transparently.

Both [140] and [143] design a name resolution system for CCN and NDN respectively that resembles DNS (maps names to records of a given type). The later uses a non routable prefix */NDNS* to reach local resolvers, i.e. like in DNS the access network provides a local cache for the name resolution service.

Concerning security, NDN relies on message signatures for security, assuming that some form of trust infrastructure is available to the communicating parties. Since name components can be content hashes, these can also be used to verify the validity of the data.

3.4.3 Network of Information

The European research project Scalable and Adaptive Internet Solutions (SAIL) [144] (following the Architecture and design for the future Internet (4WARD) project [5]) covered a wide range of services for future network including the design of an ICN architecture called Network of Information (Netinf) [138, 145].

In Netinf, names are represented as URIs in the form *ni://Authority/Local* [146, 147] where the optional *Authority* defines the scope for a *Local* part. Either part may be an arbitrary string or an hash, that includes a hash function identifier, a function id for data canonicalisation and a mimetype for the hashed content, e.g.

```
ni://tcd.ie/sha256:NDVmZTMzOGVkY2JjZGQ0ZmNmZGFlODQ
      5MjkyZDM0ZTg2ZDI5YzllMmU5OTFlNmE2Mjc3ZTFhN2JhNmE4Z
      jVmMwo:signeddata:application\%2Fjpeg
```

Netinf supports a Name Resolution System (NRS) based on an hierarchical DHT. A global NRS resolves the *Authority* component, while a local NRS handles resolution

for the *Local* part of the name. In addition Netinf also supports backends that use routing by name (e.g. NDN) as a replacement for the DHT based NRS. Instead of resolving names into locators, routing can take advantage of a different underlying architecture to route messages using these names.

The primary security mechanisms for Netinf are basic name data integrity with the use of data hashes as part of the name, and signature based integrity where the name includes a hash of a public and the data is signed with the corresponding private key.

3.4.4 Publish Subscribe Internet Technology

The EU FP7 project Publish Subscribe Internet Technology (PURSUIT) [148, 149, 150] and its predecessor Publish Subscribe Internet Routing Paradigm (PSIRP) [151] use a clean slate architecture for the Internet that completely replaces IP with a publish subscribe stack. In this architecture names are composed of multiple statistically unique identifiers, one *Rendezvous ID* and one (or more) *Scope ID*. The *Rendezvous ID* uniquely identifies a piece of data, while one or more *Scope ID* aggregate pieces of data under a scope with certain permissions.

Resolution in PURSUIT is done through a network of rendezvous nodes, implemented as an hierarchical DHT [152], where certain nodes are responsible for specific scopes. The *Scope ID* is used for internal routing in the DHT when subscribing and resolving a name. However data delivery is not routed over the DHT. Instead the rendezvous nodes involved in the process establish a route from the subscriber to the publisher to avoid the DHT routing overhead in the following exchanges.

Security may rely on content hashes as the *rendezvous ID*, but more generally it supports Packet Level Authentication (PLA) [153], that includes a signature with every exchanged packet.

3.4.5 eXpressive Internet Architecture

The eXpressive Internet Architecture (XIA) [154, 155], proposes an evolvable network architecture for the Internet. Not unlike other network protocols, it uses cryptographic hashes as network identifiers, which can be generated from either arbitrary content or user public keys.

In addition XIA provides two characteristics in its identifiers: support for principal types in the identifier and the inclusion of fallback addresses when forwarding traffic. Principal types enable the extension of addressable elements with new types, with different forwarding and addressing for the different types. To address evolution scenarios, where a new principal type is not supported by intermediate routers, XIA supports the notion of a fallback address that can be used to reach unsupported (i.e. unroutable) address types.

This means in XIA, source and destination addresses are actually graphs of different names (Figure 3.3) that express one or more ways to forward the packet. This concept of graphs as addresses is not entirely new, as it resembles source based routing, but the notion of multiple possible routes (fallback routes in dotted lines) is less common. To avoid extreme complexity the amount of fallback edges is limited. Routers can choose a path composed of the types they support to reach the destination, and the sender can include name types that are not supported by intermediate routers.

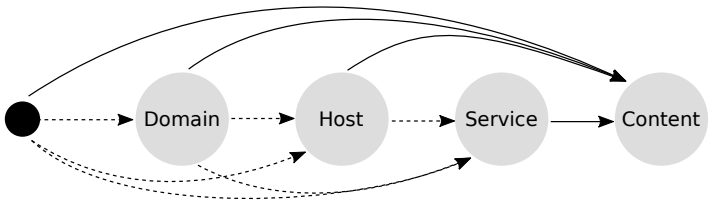


Figure 3.3: XIA destination is a graph of names

The names used inside the Directed Acyclic Graph (DAG) for (content, host, service, domain) are hashes generated from either public keys or a piece of data. XIA itself does not support mechanisms for name resolution, and it either uses an external resolution server to provide a DAG based on human input, or it relies on already existing name resolution systems like DNS. The full DAG can be seen as a list of possible names, where each full path in the DAG constitutes a variable length hierarchical name.

3.5 Implications of Naming

After an overview on existing protocols and namespaces, the implications of different namespace choices can be summarised. Along the main topics of study in this thesis, the main concern is on the security features introduced as name binding policies, and on the privacy implications of naming.

Many of the namespaces introduced previously support some form of distributed namespace that relies on cryptographic hashes of data. This is motivated for the need to introduce security functions that match the scope of the resolution system, particularly in highly distributed systems where no source of trust is available. However not all protocols support this, in particular when concerning discovery protocols. There is thus room for improvement in this context.

Concerning privacy, two forms of privacy disclosure are considered in this thesis. First the observation of network traffic and the information that can be extracted by such an observer from the names that are uncovered. And second, the ability to associate a name to a particular context through side channel attacks using other protocols

that may not even be related to name resolution, but whose subject of disclosure is a name under one of these namespaces.

3.5.1 Security challenges in name binding

A significant number of resolution systems introduce new namespaces based on some type of decentralised security mechanism to verify the resolution process [71, 125]. This is done when the resolution scope is either data that can be verified by a hash, or more specifically when it is a public key that can be used for signing the following steps.

Alternatively DNS tightens its security through the use of DNSSEC, a DNS specific PKI solution that follows the already existing hierarchy seen in DNS. Absolute confidentiality is still an issue but at least it can be addressed in some contexts.

More importantly several namespaces are actually composed from multiple other namespaces. This was already the case for URLs, but some namespaces now attempt to generalise these concepts even further.

It should be noted that among the discovery protocols that were described, composite namespaces have yet to emerge in full. Most of the described discovery systems fall under the category of insecure decentralised names, which is not surprising since in local discovery scenarios there is no central source of trust to rely on (even if some of them could in theory support this). For example, DNS-SD could reuse the techniques in GNS to include public key hashes in DNS hostnames to uniquely identify public key holders, since both use a similar namespace.

Concurrently novel netwok architectures also introduced new types of complex global namespaces. Led by ICN concepts, they propose that global namespaces should expose data identification and other types of relations by composing namespaces together. XRI allows recursive nesting of other XRI names, and the inclusion of URLs as part of a name, and IPFS also includes strategies to embed various other namespaces as part of a path. XIA introduces names as being DAGs or rather a list of possible routes.

A number of research questions are still unanswered. The primary topic of study has been routing efficiency, and the general consensus seems to be that efficient routing in these namespaces is feasible, provided routing follows network topology For example [156, 157] compares the effectiveness and costs of DONA, [158]) and one popular DHT scheme. DONA produces better results, provided the top tier routers supply significant capacity. DHTs based solutions were originally designed assuming low resource availability and high churn, where most nodes were located at the edges of the network. However DHTs with network topology awareness [159] or hierarchical DHTs with considerable resources devoted to caching can also achieve positive results [5].

For NDN, a similar approach can be followed that restricts the globally routable path labels, [140] suggests that globally routable names in NDN (i.e. the first label in a name) should be equivalent to the Autonomous System (AS), leaving the rest of the name to be handled inside the AS. As a consequence NDN names would reflect network topology, which further motivates the introduction of a separate naming system that provides topology independent names [143].

In most of these systems, message integrity is verified either through some external PKI or using hashes embedded in names to verify the associated data. However because these are routing protocols, messages are still exchanged in the clear, and visible to all entities in the routing path.

Since the trend seems to be towards variable length hierarchical namespaces composed from multiple other namespaces, names in ICN may reveal significantly more metadata about content than traditional protocols (e.g. TCP/IP headers, DNS records). In fact these namespaces have more in common with URLs than with network addresses.

The privacy implications are a consequence of the addition of more information to a global namespace. [160] points out that this has positive effects on routing efficiency (routers can apply Quality of Service (QoS) policies based on additional context) and negative effects on privacy and net neutrality (operators can throttle or sell metadata). In general ICN seems to tradeoff privacy for network efficiency [161], by providing a global namespace for content identification, with the potential side effect that it may expose more information that actually required because the name characterises content and data relations rather than hosts or services in the network. Countermeasures for these problems have been studied for in [28] and [162], and support that name privacy can be achieved in various degrees, deriving names from either a cryptographic pseudorandom function for weak privacy or encryption for strong privacy.

Notwithstanding, under these changes, it now falls onto the content producer to name content in a privacy conscious way, and consequently implement this type of solutions.

3.5.2 Privacy leakage from Names

At the network layer, privacy is often considered as a problem of data confidentiality, where a passive or active eavesdropper can inspect the packets sent by the user terminal. Such an attacker may be another user attached to the same broadcast medium, or the network service provider that routes traffic.

Historically, DNS messages are sent in the clear, and an eavesdropper can extract significant information by observing DNS messages. From the hostnames of websites

the user visits, to spurious DNS requests that accidentally disclose user web searches [105] and input commands [106].

Likewise, discovery protocols reveal information about the advertising nodes, the service it runs, or the user. This is often minimised through terminal policies that disable service discovery protocols in untrusted networks, or through specialised discovery protocols [163] that either encrypt different messages for all known recipients or through multi-party encryption for a known group.

Even so, this type of privacy leakage is restricted to eavesdroppers in the scope of the local network. The problem is exacerbated when the leaked information is globally unique, and can be used to correlate data from multiple networks. For example, terminal MAC addresses are globally unique, and can be used to track terminal movement. Conversely, very specific content names can be used infer relation between two distinct activities.

To mitigate this, pseudonymity approaches attempt to carefully manage the names leaked by network protocols and binding policies that regularly bind new names. These techniques are fairly common in user terminals for names fully or partially managed by the terminal such as IP and MAC addresses, if the terminal can bind to new names. Alternatively, privacy indirection services like a VPN or ToR can be used.

It is important to note here that naming practices similar to those seen in ICN also made their way into other architectures. For example [164] suggests the use of DNS to store named data objects. Meaning a DNS name can be resolved into a content record that includes content hashes as well as replica locations under different transport protocols. In [165] this concept is extended, allowing caching name servers to identify content names resolved by the client and provide local copies for retrieval. Conducting a search for identifiers embedded in URLs in the Common Crawl dataset [4] revealed that 112 hostnames included embedded UUIDs, and 236 included some kind of data that resembled a hash[5]. Furthermore, motivated by interoperability with DNS both GNS and IPFS (Section 3.3.3) generate DNS names and records with embedded hashes.

These issues can be partially addressed through mechanisms for DNS privacy, or through different confidentiality mechanisms such as IPSec.

In upper layers, names tend to have global scope, and carry even more data. This is widely recognised in HTTP, where the adoption of TLS has been increasing every year [166]. It is also a consequence of the widespread use of URLs to carry state information as part of their query [167]. For example, surveying URL in the Common Crawl data set for data in the various segments reveals:

[4]http://commoncrawl.org as of July 2015

[5]The search looked for known hash lengths encoded as either base32 or base16 with a minimal distribution of symbols in the target alphabeth.

- 241679 email addresses embedded in URL
- In the query component: 7397147 UUIDs and 14913196 hashes[6]
- In the path component: 2241563 UUIDs and 5106553 hashes

While introducing confidentiality addresses these problems, this is not possible for all protocols. Even when it is, alternative methods for privacy disclosure are still available, through side channels, which will be described next.

When one cannot observe the desired private information, it may still be possible to extract it from other contexts where it is used and is accidentally revealed. For this thesis in particular, the intended scope is for attacks that either exploit the handling of names, or whose result is the disclosure of a name associated with a private context.

Different protocols supply different privacy models. HTTP has a weak model with regards to client location privacy: the client network addresses must be revealed for the protocol to work, unless some indirection is introduced to prevent disclosure of this information. Other protocols boast stronger models, for example both Simple Mail Transfer Protocol (SMTP) and the Extensible Messaging and Presence Protocol (XMPP) use distributed models where intermediate servers are always used to forward messages, and in theory should conceal this type of information from the receiving party.

Either through protocol extensions or from established practices a protocol may end up disclosing information that is not strictly required for operations. Likewise the introduction of interoperability mechanisms may break privacy assumptions from one protocol as it is bridged onto another.

HTTP clients disclose the URL of the previously visited web site in the referer header [17], and SMTP servers [16]. Strictly speaking this information is not required to operate these protocols, but it was added as a feature for some parties and now a different set of stakeholders enact implementation changes to disable this type of disclosure (in browsers or SMTP servers).

Browsers in particular are susceptible to multiple types of attacks that attempt to probe the local data cache or history [168, 169, 170] to determine which URLs have been visited in the past. This in turn can be used for the purposes of user profiling [171, 172]. These issues are usually mitigated through client side mechanisms for cache segmentation. Some notable exceptions implement URL privacy policies as a service feature, are: [173] that introduces a pseudonym suffix to URLs; and [174] which studies the privacy value of URL queries over a large data set, and proposes automated methods for sanitising URLs that remove unnecessary content the query. The scheme in [135]

[6]The search looks for known hash lengths encoded as either base64, base32 or base16 with a minimal distribution of symbols in the target alphabeths.

stores user agent as part of a URL and appends signature elements as attributes in the URL query, and for privacy protection it specifically encrypts the user IP address.

3.6 Summary

This chapter went through a number of common discovery and resolution protocols, along with their associated namespaces and architectures. First, it considered discovery protocols, whose main main purpose is to provide identification within the scope of a local network. Following this, multiple name resolution protocols were covered with multiple types of security and privacy mechanisms. Finally, some novel network architectures are considered, where names offer a combination of what was seen in the previous cases, but since the namespace is also used for general purpose routing of packets its privacy exposure is even more significant.

Borrowing from the initial analogy, *we point at things we cannot name*, and some of these namespaces are composed from multiple namespaces. So a name becomes a mixture of a route that points to a destination, along with a number of assertions about the object to be resolved or one of the elements in this path. Hashes can be used to verify assertions about the resolved object, or about key holder in the path to getting the object.

The previous chapter introduced Zooko's triangle, a conjecture that states no namespace can hold all three desirable properties (security, human meaning, and decentralisation). Composition of namespaces is used to work around this limitation, by combining namespaces that hold different properties. This is not sufficient to overcome the lack of decentralised resolution (if one of the namespaces does not support it) but multiple namespaces introduce security through the addition of key or data hashes within names.

It is yet unclear how hierarchical ICN namespaces will be constructed in practice. The namespaces in NDN and CCN are a combination of globally routable prefixes and content producer defined names. A parallel can be made with URLs which follow a similar name binding procedure.

From this chapter, this thesis moves on to its main contributions towards two distinct paths. First, Chapter 4, introduces security properties into discovery protocol namespaces. These follows strategies similar to those seen in the state of the art, but then derives some additional features from consistent identification in discovery protocols. Later, Chapter 5, goes into privacy issues with different namespaces and introduces mechanisms for privacy protection with different strategies across the stack.

Chapter 4

Applying secure naming to Discovery protocols

No great discovery was ever made without a bold guess

Sir Isaac Newton

This chapter is concerned with the introduction of security semantics in existing discovery protocols. The purpose of such change is threefold. First the introduction of security semantics (as seen in HIP), provides security benefits even if the underlying transport protocol could not support it. Second, if one can embed a consistent naming scheme in different protocols, then mobility across network protocols become possible. Finally more malleable namespaces could include additional security information, including one or multiple signatures to convey trust information. These mechanisms can be leveraged further for the purposes of security, mobility and interoperability.

4.1 Introduction

Another type of device namespace commonly seen in computer networks is defined around discovery protocols. Discovery protocols provide a mechanism to advertise devices or services under a specific name. Since they rely mostly on broadcast or multicast communication as provided by the underlying network, the scope of the name assignment is only meaningful to nearby devices.

Discovery protocols are common in unstructured or ad-hoc scenarios where a central source for identification is not available. They also provide bindings between a name and other objects such as service metadata or device description. In structured

networks, the network may provide a discovery service to store and cache discovery information so the nodes are relieved from this task. This can be seen in IoT environments where nodes need to reduce power consumption.

The Host Identity Protocol (HIP)[10] introduced the notion of verifiable names to the network layer. In fact the use of a cryptographic namespace is a common solution for host mobility and security [175]. But it is most commonly seen at the network layer, to address the locator/id split in the IP protocol, or in upper layers to construct overlay networks (Section 3.3.3). However many network protocols do not have the naming flexibility necessary to use this type of approach.

Since the network address namespace will not allow for this type of approach, then perhaps going up the stack provides a namespace that does not have this limitation. In many scenarios, the next available network namespace is provided by some type of discovery protocol. This is certainly the case in IoT environments and point to point communication between mobile devices [86], where no central naming authority is available.

Expanding on this notion of embedding information within names for the purposes of security verification, it is worth considering embedding additional data within names. Hashes can bind names to a public key, but the inclusion of signatures would allow express relations of trust, or even transitive trust chains.

This chapter first approaches the use of hash based identifiers, as is common in ICN or HIP, but instead of targeting the namespace of existing or future network architectures it leverages multiple discovery protocols. From there it takes a detour to propose methods to include security information in URLs, which are a common target for proximity discovery. Since hash based URLs are already covered by existing state of the art, a more ambitious target is to include signature information in this namespace, or possibly certificates that enclose trust chains. Finally, it discusses its applications for the purposes of interoperability IoT scenarios and discusses future applications.

4.2 Establishing a common namespace over existing discovery protocols

There is a vast array of discovery protocols dedicated to enumerating network devices or services. Because such protocols cannot rely on structured networks to provide this information each network technology proposes their own, incompatible, solutions. This results in an heterogeneous environment where different discovery solutions are used for each network stack. Some of the more common discovery protocols seen in consumer equipment are the following (as discussed in Section 3.2):

- Bluetooth Service Discovery Protocol (SDP) uses its own service discovery protocol to discover nearby devices and associated services.
- Universal Plug and Play (UPnP) [176] is used in multimedia systems and home entertainment systems.
- DNS-based Service Discovery (DNS-SD) [83] is a general purpose protocol, based on multicast DNS, commonly used in consumer equipment and used in the Wifi Direct protocol stack for point to point communication between mobile devices.
- Physical systems such as QR codes, NFC, or Bluetooth Eddystone that attach digital information to an object, usually a URL to a network location.

As discussed, besides using distinct protocols, each solution employs different namespaces and scopes, some are meant for naming devices others for service identification.

- Bluetooth identifies service types using UUIDs, that are resolved into device MAC addresses/port.
- DLNA/UPnP[176] uses UUIDs formatted as URLs for service identification, these can be resolved to IP addresses/ports for each device.
- DNS-SD uses hostnames in the *.local* TLD[83], to identify both services and devices.
- QR codes, NFC, Eddystone usually store a URL that can be used to retrieve additional information from a third-party, or store application specific data as part of the URL.

Concerning their namespaces, Bluetooth and UPnP are the least flexible since they use UUIDs [177]. DNS-SD is based on multicast DNS and allows the inclusion of additional data records, such as TXT records. QR codes, NFC and custom solutions on top of existing protocols (e.g. Eddystone over BLE) identify resources using URLs with a limited size.

Like in HIP[10], the introduction of a common namespace over these protocols would enable similar features:

1. Mobility across different discovery protocols provided the terminal is multi technology (or multihomed).
2. Bootstrapping of secure transport protocols over these networks, based on hashes of public keys.

Mobility over different network architectures is a common occurrence in mobile environments. Services tend to adopt different technologies to maximise coverage over different devices and users, and likewise modern mobile devices attempt to support the features required by different services and applications (Figure 4.1). In some cases, even

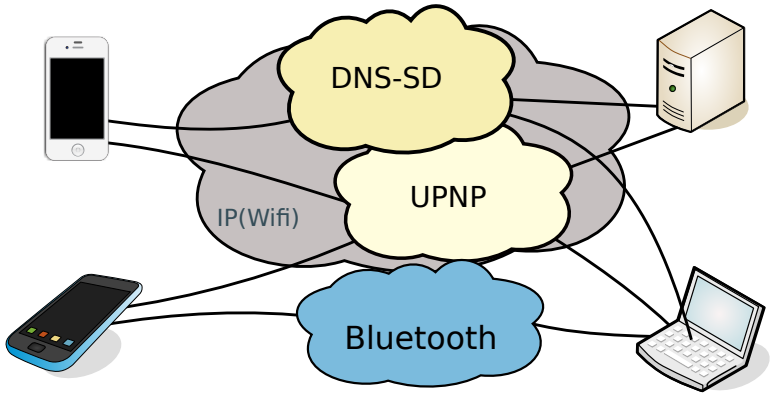


Figure 4.1: Discovery interaction across multiple technologies and networks.

if the device does not support the required stack, support can be added via upgrades or third-party applications.

Security features were not always considered when developing these discovery protocols. They defer this problem to the transport protocols, leverage models that are not always applicable (e.g. Bluetooth PIN based association), or are not always available due to device manufacturer constraints. However since all underlying network protocols provide some form of stream based communication, it should be possible to devise a transport agnostic protocol that supports public key cryptography.

To achieve the aforementioned features, the approach proposed in this thesis is to create a common namespace across all these protocols, and to define discovery functions that enable resolution of this namespace over these different protocols. An opposite approach would be to introduce a new discovery protocol that operates over all these networks, but this type of approach is not backward compatible with existing protocols. To maintain backward compatibility the proposed namespace must be representable within the namespaces for each of these protocols, that is a common subset of the namespaces for each of these protocols.

Protocols such as Bluetooth and UPnP use UUIDs as names. These have a limited size of 128 bits, with certain bits assigned specific meanings [177]. Note that Bluetooth LMP, while feasible, is not being considered here since it is mostly used for user defined names. In contrast protocols like DNS-SD, are more amenable to embedding variable length information as part of the identifier

As such one can consider UUIDs as the least common denominator. Despite their size limitations, [177] foresees the generation of an UUID from a truncated hash (UUIDv5). More generally, this approach can be applied to generate a UUID from a name in another namespace.

For this purpose a new namespace can be defined, as any UUID generated from a

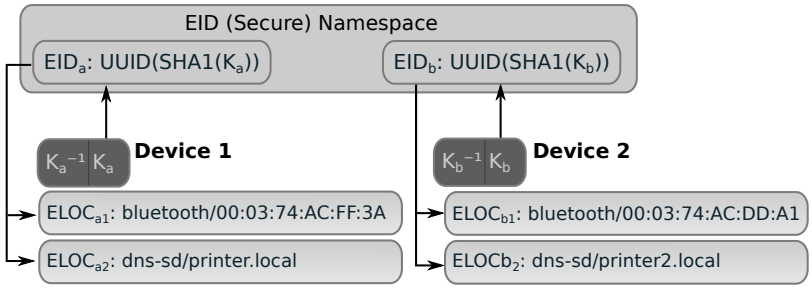


Figure 4.2: Mapping between Entity Identifiers and Entity Locators

public key. This type of UUID is referred to as an Entity Identifier (EID).

$$EID = UUID(SHA1(K)) \tag{4.1}$$

Here K is the public key in an asymmetric key pair (K, K^{-1}). According to [177], to minimise collisions with other UUIDs generated using the same method, the binding function uses a different namespace identifier for each namespace. The EID namespace identifier is *166266d3-a4b9-4886-9cb3-6d53d3928d68* (a random UUID generated for this purpose).

Each name in this context can refer to one or more devices, so the resolved objects are denominated "Entities". More generally the named entity corresponds to the holder of the private key K^{-1}, which can be one or multiple devices.

In practice, since the namespace is being resolved using network discovery protocols, the resolved objects are in fact network locators (Figure 4.2) or Entity Locators (ELOCs). Regardless of the wire format, the resolved ELOC always includes a protocol family that identifies the protocol in use and a locator address:

$$ELOC = ProtocolFamily/ProtocolLocator \tag{4.2}$$

4.2.1 Discovery functions

Around the EID namespace, one can define a protocol agnostic discovery framework that applications can call upon. The goal is not to replace existing discovery Application Programming Interfaces (APIs), but rather to complement them with additional functions. This framework does not impose a particular model on the underlying discovery protocol. Both centralised or distributed discovery protocols can be supported, provided an UUID namespace can be represented in the underlying namespace.

Four core functions define this framework, and correspond to the API made available to the applications that use the framework:

Publish Entity(EID, [ProtocolFamily ...]). Create the protocol specific identifiers, and advertise our own entities, by publishing EID information to the network, allowing other devices to discover them.

Discover Entities([ProtocolFamily ...]) $\rightarrow [(EID, ELOC)]$. This function is the general enumeration function, that discovers nearby entities and maps them to the appropriate locators. Each discovery result take the form of a tuple *(EID, ELOC)*, that maps the discovered entities to locators in specific protocol families.

Entity-to-Locator(EID, [ProtocolFamily ...]) $\rightarrow [ELOC]$. Returns the locators for the entity that advertises a specific EID. This can be used to find locators for entities that have been encountered previously, and are known to hold a specific EID.

Locator-to-Entity(ELOC) $\rightarrow [EID]$. Translates a locator from a specific discovery protocol, to an EID. For example this is used to translate between a device identifier (such as a MAC address), to an EID. A locator may have multiple associated EIDs.

From these functions one can quickly support two use cases: first aggregation of discovery result from multiple protocols; and second mobility across protocols. The optional $ProtocolFamily$ argument enables the caller to specify which protocol should be used when calling protocol independent functions.

Discovery aggregation (fig. 4.3) allows an application to discovery the same entity over multiple protocols, with each discovery protocol in handled internally by the discovery framework. Based on this information the application can then decide which protocol is more suitable. For example, Bluetooth might be more energy efficient for small payloads, but for transferring larger payloads WiFi may be preferable due to higher throughput. After finding the desired EID, the application can connect over the chosen protocol and start service authentication based on the EID.

At a later time, if a new discovery protocol (C) becomes available, then the application can re-discover the previous EID in this new protocol (fig. 4.4). Afterwards a new session can be established over the new protocol, requiring a reauthentication for that particular EID. If the underlying transport protocol supports it, then techniques for session resumption (e.g. [178]) can be applied to avoid a full reauthentication exchange.

From a different perspective, this is essentially mobility across different transport protocols. The discovery framework provides the necessary information to enable this, mapping locators from multiple network protocols, but the application (or some additional middleware) still needs to handle the choice of protocol and authentication.

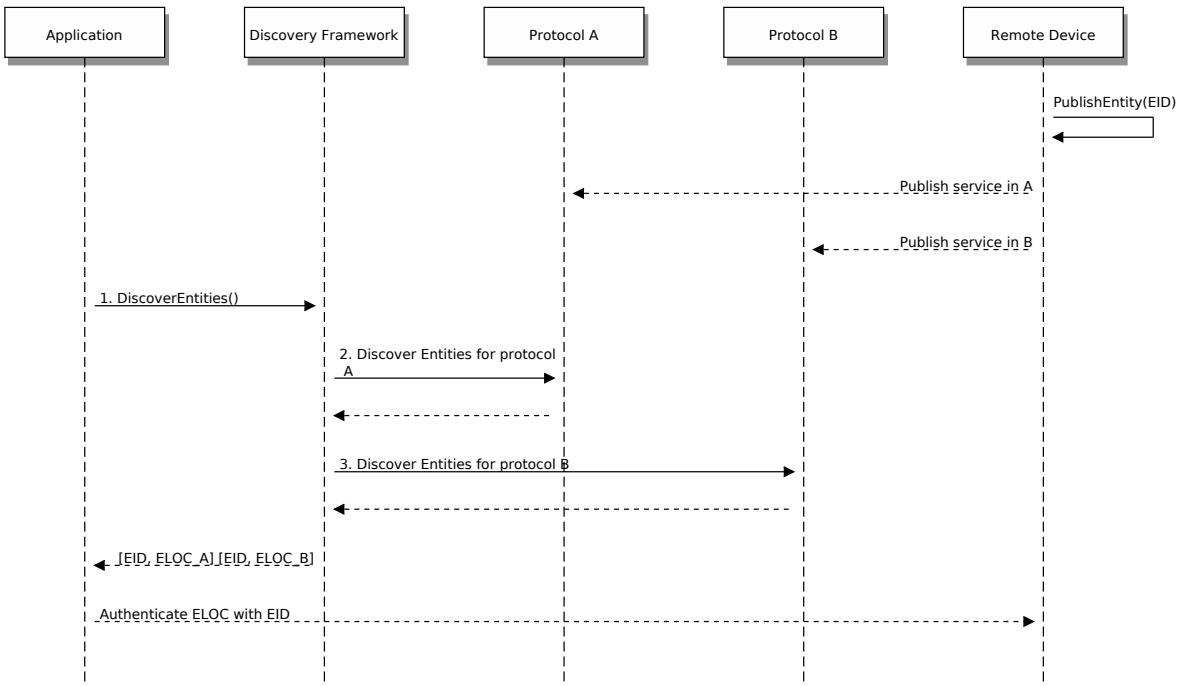


Figure 4.3: Discovering all entities for two protocols A and B

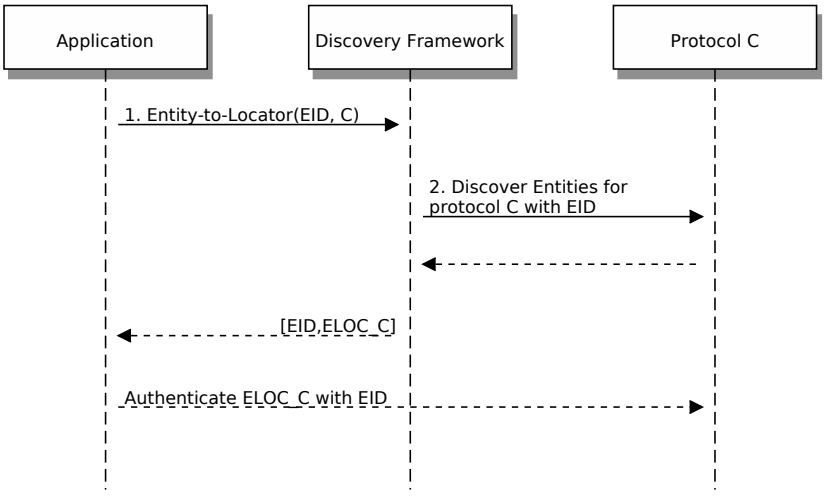


Figure 4.4: Find a locator for a known EID in a specific protocol*(C)*

4.2.2 Instantiation

With the proposed framework one can reason about discovery processes as composed by the described functions. For experimental purposes this framework was implemented on top of four existing discovery protocols:

- Bluetooth SDP (through the Bluez bluetooth stack)
- DNS-SD (through the bonjour libraries)
- UPnP (using the Coherence implementation)
- Named Publish Subscribe Networking (NPSN), a centralised discovery protocol for NDN

Naturally not all these protocols hold a one to one mapping with the described functions. The purpose of this approach is to retain backward compatibility, keeping protocol messages unchanged, while overloading existing messages to implement the aforementioned functions. NDN is included here as an example of a non legacy system, through NPSN, a custom publish/subscribe discovery protocol designed for IoT environments [39].

For Bluetooth, an additional SDP service is announced (with the desired EID). Discovery of an EID (a device, or group of devices) is carried out as searching for a service type with a specific UUID. This corresponds to a change in semantics. In Bluetooth SDP this process was originally meant to query for general service types (e.g. printers, sound sinks), while for an EID aware application the same protocol exchange means to query for the holders of an EID. A similar approach is also followed for UPnP, by using the EID in the USN field.

In DNS-SD, discovery is scoped per service types (e.g. _http._tcp for HTTP servers), and devices are enumerated for each service type. For publishing an EID, a record is published under the *_uuid._tcp* service type. Under this service type, devices advertise with their EID as a service name, that can then be mapped to a hostname. This is also a common approach used by other applications to advertise additional identification information in DNS-SD.

NDN lacks intrinsic discovery mechanisms, but different discovery protocols are already emerging on top of this stack. For example, throughout this PhD two different discovery protocols were proposed for NDN, one for ad-hoc scenarios [40] and another for publish subscribe environments [39]. The later protocol, called NPSN, was intentionally designed to include the scheme proposed in this section as a mechanism to facilitate interoperability, with other discovery protocols. Because NDN network identifiers are variable length hierarchical paths, a generic approach is to include the UUID in one of the labels of the path. NPSN is centralised, in the sense that the local

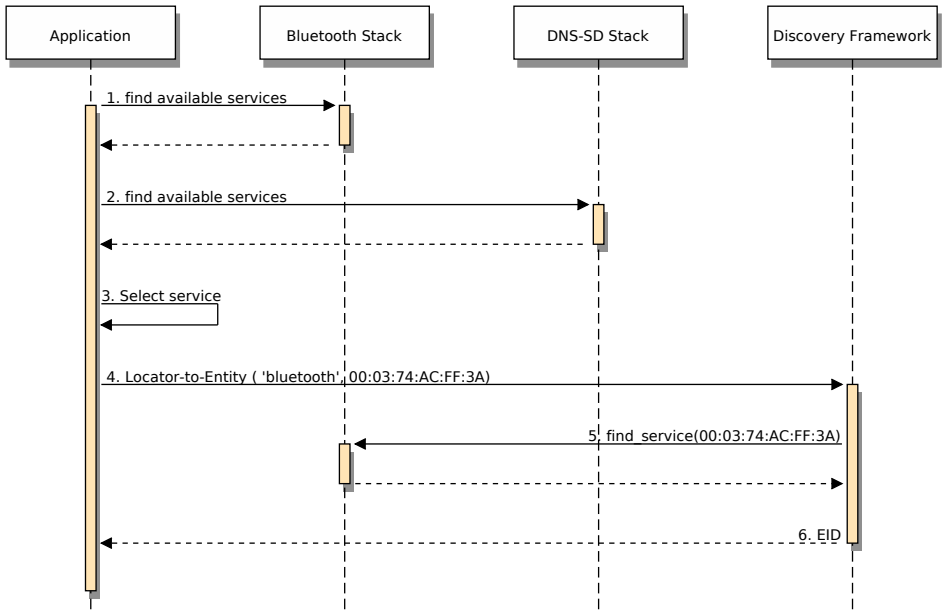


Figure 4.5: Discovering devices and the associated EID

network provides a rendezvous server to cache and aggregate discovery information for local nodes. This shifts work from the clients and services onto the rendezvous server.

Initially this framework was implemented as a general purpose library [1] for cross protocol device discovery [38] over Bluetooth, DNS-SD and UPnP. Its main purpose was to provide a client device the ability to rediscover known services while switching the underlying communication protocol (cross protocol mobility) while minimising service requirements other than the initial configuration. Internally, this framework is available as an API used by applications, in combination with the regular APIs from the supported discovery stacks.

For example, in its initial state (Figure 4.5), an application has no information about services in the network and performs discovery using available protocols. It then follows a series of steps to select a suitable service and determine its EID:

1. The application discovers services available using Bluetooth
2. Concurrently it also discovers services available using DNS-SD
3. The application selects one service instance e.g. 00:03:74:AC:FF:3A discovered using Bluetooth
4. The application calls the Discovery Framework to determine the EID associated to that particular device.
5. The discovery framework does an additional Bluetooth query to determine the EID.

[1] Publicly available at https://github.com/ATNoG/python-udiscovery

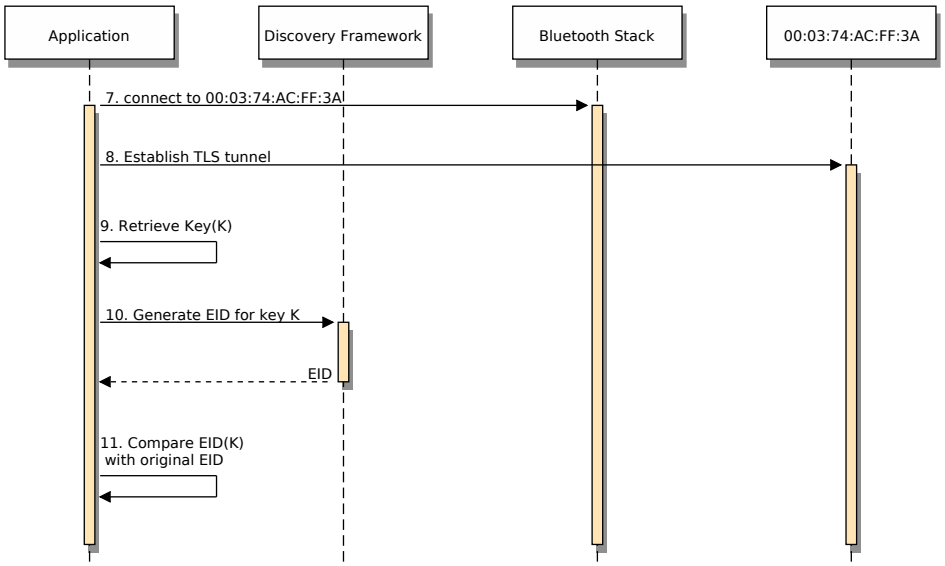


Figure 4.6: Authenticating known devices

6. If the device has an associated EID, it is returned to the application

Upon completion the application holds both a network locator and service metadata, as well as an EID. With these elements the application can start a new connection to the service, using the EID to verify the authenticity of the connection.

Now that it was established how to discover devices based on the EID, they can be combined with other protocols to enable device authentication. Assuming the EID is generated as a hash of a public key, one can establish a connection using protocols such as TLS that rely on public key cryptography.

Continuing from the previous step, after discovering a specific EID and a locator an application would (Figure 4.6):

7. Use the ELOC from the previous step, to open a connection to the remote device.
8. Negotiates a TLS tunnel with the remote party
9. When the tunnel is established, the application extracts the remote party public key
10. Generate the EID from the public key
11. Finally the EID for the retrieved key is compared with the original EID.

If both EIDs match, the authentication succeeds i.e. the remote party is in possession of the corresponding private key, and communication can proceed.

When, at a later time, the same application attempts to reconnect to the holder of a previously used EID (Figure 4.7) using the same or a different protocol, it can discover the locators for that specific EID:

1. The application queries the discovery framework for a device with the given EID

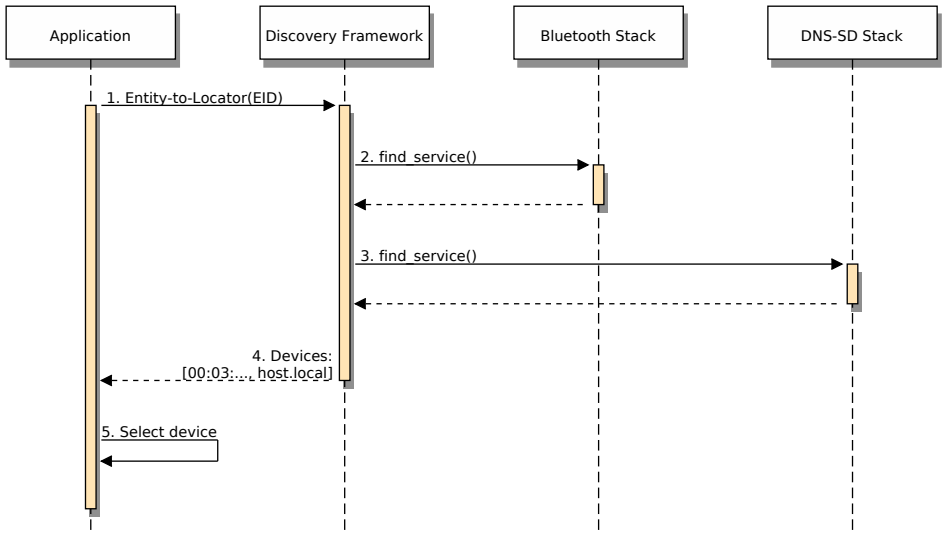


Figure 4.7: Reconnecting to a service using its EID over multiple protocols (Bluetooth and DNS-SD)

2. The discovery framework queries for Bluetooth devices with the given EID
3. The discovery framework queries for DNS-SD devices with the given EID
4. The framework returns a list of all locators (with the given EID) found for both protocols
5. From the returned device list, the application selects one, e.g. host.local over DNS-SD

It is important to realise that the subtle change in semantics for some of the protocols (Bluetooth, UPnP) does carry implications. Because device enumeration (i.e. EID enumeration) is in fact service type discovery, this implementation always requires an additional round trip time for Bluetooth SDP (step 2 in Figure 4.7 to find the EID for a device). Likewise, for DNS-SD, an additional query might be required (or an additional subscription at initialisation). However the additional costs can also be amortised through caching, which is already used in various protocol stack implementations. Bluetooth stacks rely heavily on result caching, for a fixed period of time. DNS-SD stacks use a publish/subscribe model where a background service handles updates, thus exhibiting constant times (near zero), because the results are already in cache.

In both DNS-SD and NPSN there is an additional cost for framework initialisation. DNS-SD requires an initial subscription to be made at startup, and a NPSN session needs to be created with the rendezvous. In a Linux machine with a 1.3Ghz processor this time averaged to 210 microseconds (for 1000 repetitions) to initialise the library. However it should be noted that the DNS-SD libraries are most likely delaying

initialisation, making it hard to measure the actual time cost.

While the previous examples (Figure 4.5) represents discovery operations as sequential, the implementation does in fact take advantage of parallelisation of the different discovery protocol backends.

4.2.3 Security Considerations

This framework ties public key cryptography authentication with existing discovery protocols. However there are some limitations that fall out of scope of these changes and should be made clear.

For security purposes, the *EID* is generated from a hash of a public key, and the desired security capabilities are a consequence of this property. It is assumed the caller would take advantage of the EID namespace (Equation 4.1) to authenticate public keys used by services, for example when using transport protocols like TLS. Other generation methods can be considered under a different security model. The input for UUID generation is always a hash, but it could be the hash of any piece of data. For example UUIDs could be generated from DNS domain names [179], Uniform Resource Identifiers (URI) [55] or X509 Distinguished Names [180]. Such change would require other mechanisms to authenticate services, one common case being the Public Key Infrastructure (PKI) used by clients that use TLS.

The described framework introduces names whose purpose is to be used for authentication after discovery. However, since the security properties of the underlying discovery protocols remain unchanged, traditional repetition attacks against these protocols are still viable. Malicious nodes can forge messages (e.g. a replay attack for a known EID) that would work as a Denial of Service Attack (DoS) attack.

While the EID can be used for authenticating a key holder, it cannot assert trust. The caller should store the EIDs for services it considers trustworthy or at least for pinning services it used previously.

Ultimately the EID is a hash of the key and collisions might occur, by coincidence or as a generated collision, and SHA1 is known to be vulnerable to attack [181]. Other hash functions are available as replacement, but due to size constraints the hash would need to be truncated to fit in the UUID. Ideally the client should store the keys associated to the EID.

4.3 Building interoperable discovery gateways

The variety of different discovery protocol, results in an interoperability problem when attempting to interconnect these networks and devices. Even if the communication protocols in use are the same, no communication is possible if discovery is not possible.

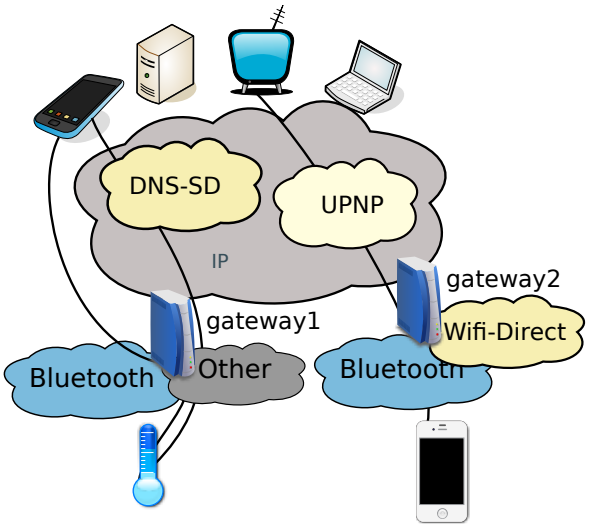


Figure 4.8: IoT scenario involving multiple services, consumers and gateways

One field where interoperability issues are common is IoT (Figure 4.8). The large number of incompatible technologies and vendor specific protocols which is an important scenario for our concerns leads to the introduction of compatibility gateways, that provide protocol conversion services. These gateways retain the necessary state about network devices, forward discovery information across different protocols and convert protocol messages between devices if necessary.

The common identification scheme seen previously (in Section 4.2) establishes a global discovery namespace. Intuitively, a global namespace would facilitate service gateways deployment, since gateways could advertise the same identifier over multiple discovery protocols and avoid state retention. However the namespace from Section 4.2 is only used as a global namespace that is resolved to the objects provided by the underlying protocol. Before it can be used in IoT scenarios, additional challenges need to be identified and addressed to build interoperability gateways that leverage this namespace.

Within the context of IoT interoperability scenarios (Figure 4.8) challenges arise as a consequence of the use of multiple protocols and intermediate gateways. To break apart these challenges, the illustrated scenario can be described as four distinct basic cases:

Case 1: When considering a simple IoT application, a representative scenario starts as an set of sensors (Figure 4.9), that expose their services using discovery protocol $P1$.(e.g. Bluetooth or Wifi Direct). The discovery protocol fulfils the role of finding the services associated to those sensors, but it is possible the client only wants a specific sensor (S1), or a subset of all sensors (e.g. S1+S2 but not S3).

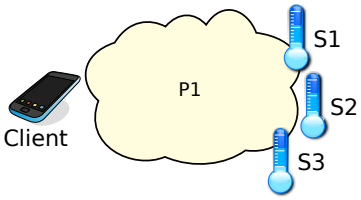


Figure 4.9: Case 1: Communicate with (sub)sets of sensors

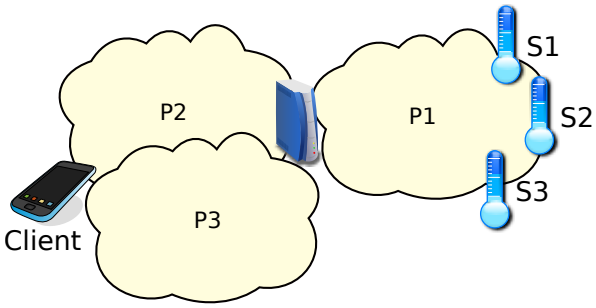


Figure 4.10: Case 2: Extending discovery to other protocols using gateways

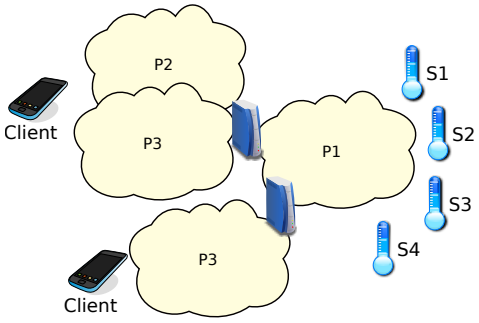


Figure 4.11: Case 3: Gateway replication

Case 2: To extend the previous scenario, and allow exposing the sensor(s) onto other protocols, a gateway is added (Figure 4.10) that advertises the services over two new protocols ($P2$ and $P3$). This opens up possibilities for clients that support multiple protocols to take advantage of intelligent strategies, switching protocols as they perceive to be beneficial. However this can only be achieved if they hold the capability to identify both services being advertised as being equivalent.

Case 3: The introduction of additional gateways, will widen the coverage range for one of the protocols or provide redundancy (Figure 4.11), but exacerbates compound on the challenge presented in case 2. The problem of introducing multiple gateways that map onto the same protocol ($P3$) is the risk of mapping the same service/sensor multiple times, potentially confusing the clients on that protocol.

One solution for this issue would require gateways to communicate among themselves to exchange service rosters, and avoid duplication of service advertisement. However, this kind of approach creates a strong emphasis on the knowledge detained by each gateway, and on gateway coordination. In decentralised scenarios gateways cannot assume full connectivity between gateways or that all gateways fall under the same administrative domain.

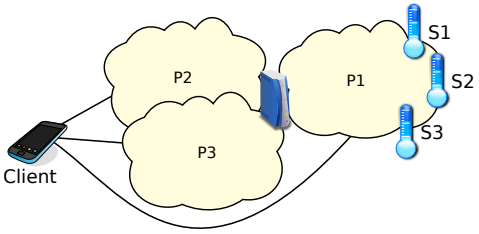


Figure 4.12: Case 4: Discovery consistency across gateways

Case 4: One final case (Figure 4.12), derived from $Case2$, is when the client supports multiple protocols (and technologies), one of those being the protocol used by the sensors, in which case the client may bypass the gateway entirely and be presented with a service it cannot identify as being the same. While it might seem the gateway is unnecessary (for that particular client), $P2/P3$ may provide wider coverage than $P1$. More importantly the client should be able to realise that the services being advertised over protocols $P2$ and $P3$ (i.e. advertised by the gateway), are the same services being advertised over $P1$.

The namespace implemented in Section 4.2 is a good fit for these features. Multiple devices can advertise the same EID ($Case1$), and the client can then choose a specific sensor or multiple sensors. Assuming the gateway publishes discovery information while retaining the EID the remaining cases are also be covered. Since the EID is consistent in all supported protocols, clients can identify services discovered using distinct protocols as being the same service ($Case2$) and choose the preferred protocol. Likewise, gateway replication ($Case3$) will cause identical services, with the same EID to be published by multiple gateways, even if the resolved information refers to distinct gateways. Finally $Case4$, can be seen as a variation from $Case2$, where the client supports the same two protocols as the gateway.

An IoT gateway that bridges two protocols can be seen as a node that performs the following actions:

1. Discovers services advertised by sensors using the source protocol
2. Advertises matching service discovery information in a target protocol

3. Translates data requests between the target and source protocol, i.e. conversion of data representation from sensors in one transport protocol to another.

In [41, 39] the EID namespace is used to introduce decentralised IoT gateways, with data conversion between Message Queuing Telemetry Transport (MQTT)[182], Constrained Application Protocol (CoaP)[26] as well as NPSN [39] and PURSUIT [148]. Other data conversion approaches can be found in the literature for other protocols [183, 184].

However, before this framework could be adopted for this type of scenarios two shortcomings needed to be addressed. First the implementation lacked any consideration for transport ports (as seen in UDP, TCP or Bluetooth RFCOMM) i.e. the EID identifies one or more devices but cannot identify an upper layer protocol endpoint.

Thus the ELOC is extended from its original form (eq. (4.2)) to a three element tuple with an optional element designated *ProtocolChannel*.

$$\begin{aligned} ELOC &= ProtocolFamily/ProtocolLocator \\ &\quad [/ProtocolChannel] \end{aligned} \tag{4.3}$$

The *ProtocolChannel* holds the service information as returned by the underlying discovery protocol. This element is optional to retain compatibility with the case where the EID does not map into a service, or the discovery protocol does not provide one. For Bluetooth the *ProtocolChannel* holds the Bluetooth channel, for IP based discovery the TCP or UDP port. In NPSN, it is absent since the NDN name already includes all necessary information.

A second limitation is that the EID binding always associates an EID with a public key i.e. it always identifies the owner of the key. However clients may be looking for opportunistic service with no expectation of trust, for example the nearest sensor with a certain service type. In other words, it would be desirable to have a cross protocol namespace for service type identification.

The binding method used for the EID [177] could accept an arbitrary piece of data as argument, however the resulting name can not be authenticated when establishing a connection. Reusing the same technique, the binding function for a Service Type Identifier (SRVID) namespace can be defined as

$$SRVID = UUID(SERVICE\_ID\_STRING) \tag{4.4}$$

The resulting name is still an UUID but is meant as an identifier for specific service type. To minimise collisions with the EID namespace a new namespace identifier is used *957d7f49-2674-4e12-ab92-4a1678d82d03*. To generate an SRVID from a service

type, the binding function as stated previously is applied using a service type as the content argument, e.g. for the string *"printer"* the resulting identifier is

$$47ac0c1e - e377 - 5b5d - 92cf - 18b489835781 \tag{4.5}$$

As the hashing operation is non reversible, i.e. it is impossible to get the human readable service type from the hash, a similar requirement to other discovery protocols that use UUIDs is to keep a mapping table of well known service types into matching identifiers (e.g. UUIDs or ports). For now, it is left up to the caller to choose the source material for the SRVID. Different network stacks use their own canonical centralised namespaces for this (e.g. IANA service names for IP services, or Bluetooth known service type UUIDs).

Since any device can advertise multiple EIDs, consequently it can also advertise as many service types as necessary, as is the case for most service discovery protocols. With this information in hand, a client can cross reference discovery results to find a device that simultaneously advertises the EID for a known key and a SRVID. This yields both a service locator and a method to verify authenticity based on the public key. Naturally this comes at the cost of additional discovery messages. The messages exchange from Figure 4.4 needs to be repeated from each EID/SRVID.

An alternative method would be to generate a namespace that combines both the hash of the public key with the hash of the service type, e.g.

$$EID^* = UUID(SHA1('printer') + SHA1(K)) \tag{4.6}$$

However this approach would prevent the clients from discovering devices based on service type alone, or the sensors would need to advertise even more services for all possible combinations which is also undesirable in constrained scenarios.

4.4 Embedding security information in discovery URLs

The previous section discussed the construction of a verifiable namespace over UUIDs, that worked through the introduction within the UUID of hashes constructed from a public key. From here, the next target in this type of protocols is not the namespace but rather the scope of the protocol which is often a URL.

Due to size limitations, URLs are not used directly in the protocols discussed earlier, but they are commonly used in other discovery contexts (Section 3.2.4), not as names but as objects in the resolution scopes, which are then used as locators to a service. Because URLs are composed from multiple namespaces, it is common to use hashes as part of an URI. Some schemes, like magnet links [136] and the named information

scheme [59] were designed solely for the purpose of generating location independent URLs for pieces of data.

In proximity scenarios, service discovery can be triggered from scanning an NFC tag, QR bar code or based on other lightweight protocols (e.g. Eddystone). These often include a URL that locates the contact point for the service.

This type of scenarios motivated the work done in [36, 37], an electronic ticketing service for mobiled devices, with mutual authentication between the service provider and the ticket holder, this leverages this same approach. The primary goal was to provide mutual authentication mechanisms in mobile scenarios, even when global connectivity could not be assumed. This often involved authentication procedures that started from unauthenticated discovery information using one of the protocols described earlier in Section 3.2.

Given the different types of mobile devices, multiple methods mechanisms for initialising a process from proximity discovery are available. In [36] multiple types of scenarios were supported allowing service discovery to be started from three sources:

1. QR codes
2. NFC tags
3. Bluetooth

The result of the discovery step is always a URL, used to reach the ticket consumption service, through one of the supported transport protocols Ticket submission can occur either over Bluetooth or IP (i.e. Wifi or WifiDirect [86]). Internally, ticket generation and submission uses a Representational State Transfer (REST) webservice over TLS.

In [36] each entity (client, seller, consumer) holds a public/private key pair. Tickets (Figure 4.13) are signed by the seller and include the public key of the ticket holder (B_P), a seller certificate (V) and a consumer certificate (E). This is sufficient information for the client to authenticate the ticket consumer service and vice-versa.

Ticket generation and consumption is illustrated in Figure 4.14. The user equipment acquires a ticket from a seller, generated from a private key B_P from the user's device (steps $1-3$). Each ticket is generated from a new user key, to avoid identity correlation from multiple sellers. For ticket consumption (steps $4-8$), the user, starts by discovering a nearby ticket consumption service and establish a connection. From the discovered information the user can also determine which tickets can be used with this service. The certificate E is used to determine if the TLS connection should be trusted before submitting the ticket, this avoids disclosing the information in the ticket to untrusted third parties. The ticket consumer service authenticates the client as the

Figure 4.13: Internal ticket structure for Multipass

correct ticket holder through the key B_P (used to establish the TLS connection), and verifies the ticket is valid through the seller signature $Sig(V_K)$ and certificate V.

To avoid the introduction of fake discovery tags in step 4 (Figure 4.14), all tag contents are signed by the ticket consumer. The URLs inside the tag contains the following items

1. The original **URL**, pointing at the intended service
2. An **identifier** used to identify the signing party and/or signing method
3. A cryptographic **signature** of the previous two fields

The original URL is separated from the remaining fields with the # character, while the identifier and signature are separated by a question mark (?), e.g.

```
https://atnog.av.it.pt#keyid?XAOyG28RNDKDdZW4as3diAuiDXTXz
ZiZsyV_Znvx3b1HjA8aVHvvAbkzcVc0FDzw
```

Choosing # as the separator character ensures the resulting representation is itself a valid URL, with the key identifier and signature encoded in its fragment. In HTTP, URL fragments are used as local subresource identifiers [55, 185], and are never sent to the remote HTTP server. This guarantees backward compatibility, in the sense that the URLs encoded with this method can continue to be used as regular HTTP URLs. If the application reading the tag understands the additional semantics then it can verify the signature, otherwise it can use it as a regular HTTP URL.

Other alternatives, such as [135], encode signature attributes as part of the URL query. This makes them more generic, since they can be applied to any URI scheme, but will send the entire signed URL when making an HTTP request.

The signature sizes vary with different encryption algorithms, and key sizes (Table 4.1). To include the signatures inside the URL these need to be encoded in base64 [130], which causes an overhead up to 33% additional bytes. The values presented in Table 4.1 were obtained from the DER encoding used by the OpenSSL toolkit[2]. For even shorter signatures there are variants of the DSA and ECDSA [186, 187] that can make signatures up to 25% smaller.

[2]https://www.openssl.org

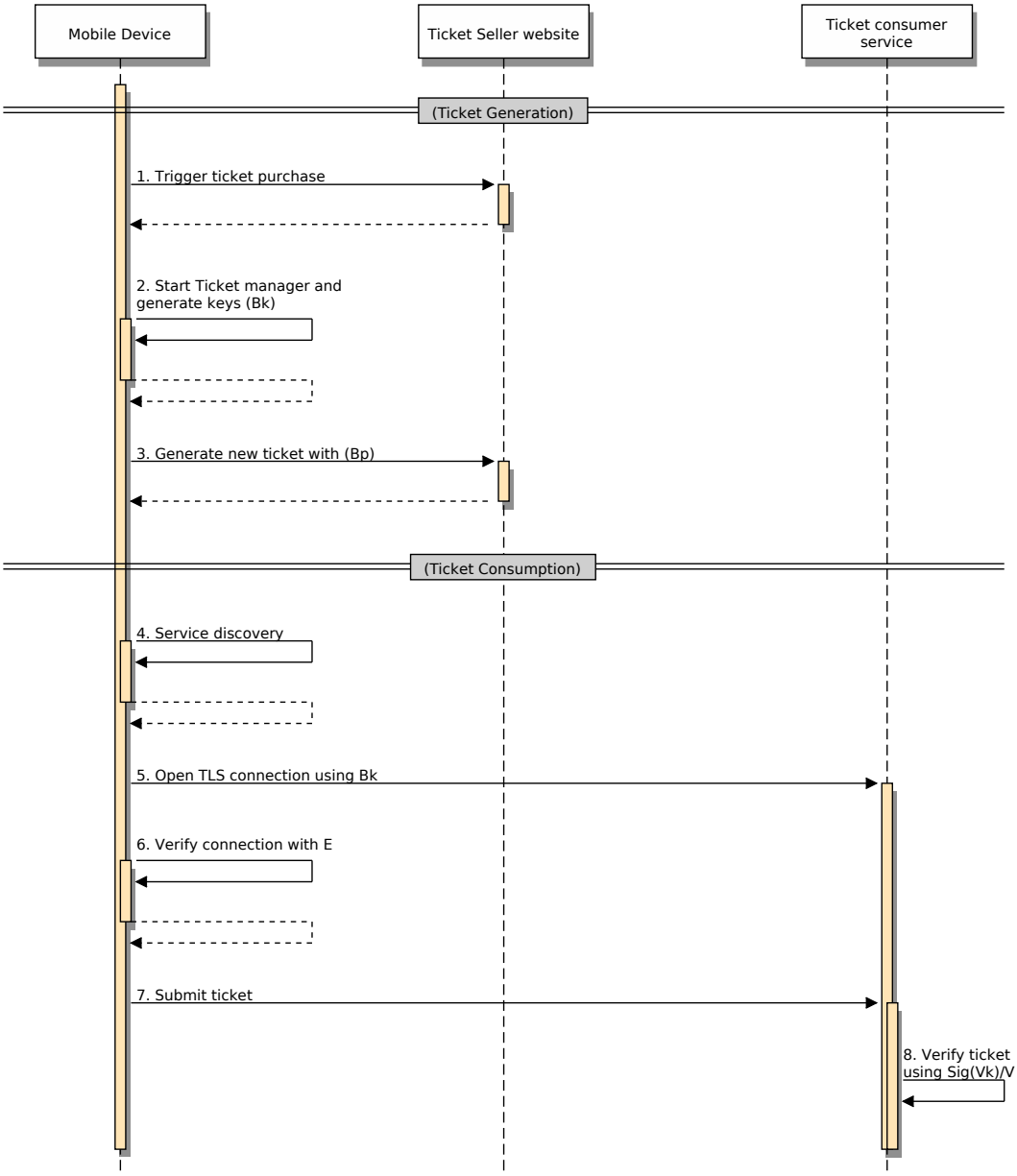


Figure 4.14: Ticket generation and consumption

Table 4.1: Signature length for multiple algorithms

Algorithm	Signature length (bytes)	Base64 length(bytes)
ECDSA secp256k1	71	98
ECDSA secp512r1	139	191
DSA	71	98
RSA 2048 bits	256	349
RSA 4096 bits	512	693

Table 4.2: Key/Certificate length for multiple algorithms (base64 encoded)

Algorithm	Key length (bytes)	Certificate length(bytes)
ECDSA secp256k1	174	420
ECDSA secp512r1	268	611
DSA 2048 bites	1198	1448
RSA 2048 bits	451	964
RSA 4096 bits	800	1655

With base64, signatures can be included in URLs as described earlier, the largest signature (RSA 4096 with 693 bytes) would still be within the maximum size of a QR tag (2953 bytes). However, for pragmatic reasons described in Section 3.3.5 the maximum size of an URL, can be considered to be 2000 bytes.

Attempting to include additional information, such as a public key or certificate may run into the maximum length. Elliptic Curve keys are much smaller in size, and make it possible to encode certificates inside the URL, which opens the door for delegation, as is done inside the tickets (Figure 4.13).

Table 4.2 shows the key and certificate sizes (base64 encoded) of different algorithms. The certificates in this example are encoded in X.509 [188], and include only the minimal required information. An alternative certificate representation format is specified in [189], which is also used in NFC signed records. This can save up to 40% (60% with optional optimisations) for the elliptic variants [190].

The elliptic curve variants are the more promising in terms of space (Table 4.3). For example the ticket format depicted in Figure 4.13 (one key B_P, two certificates E, V and a signature) would require at least 1112 bytes to be encoded inside a URL with the smaller curve. From the other algorithms, only the RSA 2048 is small enough to fit in a QR code, but possibly too large to be used in an HTTP URL.

Finally, as pointed out in [94], even if data in these tags is signed by a trusted party, it is still possible for an attacker to implement replay attacks, where the correct tag is replaced with another valid tag.

A valid, but expired, tag can be used as a DoS attack, leading them to a service that will never respond. When using passive tags (i.e. QR codes or passive NFC tags)

Table 4.3: Encoding size for a full ticket (base64 encoded)

Algorithm	Ticket length (bytes)
ECDSA secp256k1	1112
ECDSA secp512r1	1681
DSA 2048 bits	4192
RSA 2048 bits	2728
RSA 4096 bits	4803

a DoS attack cannot be avoided, if the attacker can tamper with the tags.

A valid tag from a different physical terminal can be used to to deceive users into buying the incorrect items[94]. To avoid this, full authentication requires an additional step where the physical device in proximity with the user provides confirmation before authentication can proceed. For example, [191] generalises this notion to incorporate radio interference into the authentication process to determine if a MITM attack is occurring.

4.5 Conclusions

Generally speaking, this chapter introduces techniques to embed security information in different types of discovery namespaces, and details this for UUIDs and URLs. UUID based namespaces are used is common discovery protocols as service identifiers (Bluetooth SDP, UPnP, DLNA), and other protocols that fulfil similar roles can accommodate the inclusion of an UUID as part of their names.

The advantage of implementing this as information embedded is discovery protocol identifiers is that no changes to the network or even discovery stacks are required. These approaches can be implemented on top of existing discovery protocols, for example as part of application logic in mobile devices, with no need for low level changes to either the network stack or the discovery stack.

In total, three name binding functions are defined in this chapter, the EID, SRVID and URLs with embedded signatures and certificates.

The EID is a namespace for device identification based on hashed public keys. It is heavily inspired by the HIP protocol and it is used to established authenticated connections over multiple protocol stacks. For each of the considered protocols (SDP, UPnP, DNS-SD, NPSN) this is achieved by compressing the desired hash as a UUID and providing discovery functions between this UUID and the protocol specific network addresses. In a way, the EID can be considered to be as a overlay discovery protocol. Currently, the list of supported protocols covers the majority of consumer equipment protocol stacks and it seems likely others could be added as well.

The SRVID is nearly identical to the EID, but it foregoes the authentication based on public key for a simple identification of service types. It is used for advertising network protocol independent services that can operate over different protocol stacks. It does not provide authentication, but it can be combined with the EID to achieve both these properties at the same time. However this requires a network agnostic transport protocol that can use public key cryptography (a role fulfilled here by TLS).

With the provided discovery functions, applications can implement cross protocol session mobility, moving over to the protocol that offers the best energy/bandwidth efficiency or based on other strategies. Furthermore, nothing prevents the keys used in other protocols (e.g. HIP) from being used in an EID or other similar systems, at which point it is possible to integrate with HIP based resolution or other upper layer protocols based on overlay networks.

The URL embedding strategies (Section 4.4) are meant for discovery operations that use URLs as general purpose locators in discovery scenarios. In particular they were used to increase resistance to interception attacks in a mobile ticketing service. This is achieved through the inclusion of signatures used to validate discovery information based on pre-stored keys, or alternatively one or more certificate to express trust relations and delegation for signature verification. However, inclusion of certificates quickly reaches the limitations imposed more constrained URL storage media, such as QR codes or practical implementation limits

Ultimately these mechanisms do not alter the security properties, (or lack of security), since they remain vulnerable to replay attacks, at least as a form of Denial of Service Attack (DoS).

These mechanisms are partially composable, as one can include an EID as part of a URL. However the reverse is not possible due to size constraints. A straightforward conclusion is that one cannot expect to rely on external signatures to detect fake discovery information when crossing those networks, i.e. anyone can announce any identifier. The implementation approach that was taken is heavily geared towards backwards compatibility with existing devices and services. Section 3.2.4 already introduced alternative protocols in Bluetooth that are based on URL advertisement, but the size constraints remain. If the need arises, then new extensions would need to be introduced into the stack of these protocols.

The following chapter will continue onto a different path, that of privacy in global namespaces. While this topic may be claimed to be orthogonal to the work done thus far, this chapter spent a significant amount of effort embedding different identifiers on top of often insecure discovery protocols. This approach is not dissimilar to the one seen in other namespaces covered in Chapter 3. However as these namespaces become global, which is a likely consequence of the IoT mapping mechanisms introduced here,

then privacy becomes increasingly relevant.

Chapter 5

Realising Namespace Privacy

Any privacy in public is a hard thing to negotiate

Benedict Cumberbatch

In this chapter, the main subject is that of naming privacy and the privacy mechanisms implemented in this thesis. First it introduces pseudonymity solutions at the network layer. Following this, it goes up the stack, first to identify sources of privacy leakage at the upper layers and then to propose a service enforced privacy namespace for URLs.

5.1 Introduction

Digital privacy is now a common topic. Society is presently engaged on a debate surrounding privacy and its relation to values such as freedom of speech, security and the role of technology. Much of this debate has been centred on user privacy, and data disclosure, but privacy is not an absolute concept but rather dependent on context[192].

Names, in this context, are a fundamental form of metadata, a form of identification for various types of resources in the network. In the previous chapter, as in much of state of the art covered in Chapter 3, a general tendency towards namespace composition can be observed, in which new namespaces are created from previous ones. In other words, based on ongoing research trends and empirical observation, the amount of information placed in names seems to be increasing. Consequently disclosure of this type of metadata becomes more revealing, and depending on the specific name it may reveal location, user identity, relations to specific content, or trust in some authority.

Initial contributions to this topic aimed to introduce support for privacy mechanisms related to network exchanges. In particular using three types of mechanisms to achieve this goal: first on the client side (i.e. in the user terminal equipment) through

pseudonymity solutions, second with support from network infrastructure, and third through the integration of IdM mechanisms that apply pseudonymity with regards to external services. This is the work described in Section 5.4

A realisation that followed from this work is that the previous techniques can only apply pseudonymity to those namespaces that are "close" to the user, in the sense that the user is involved in the assignment of names in this namespace or is directly connected to those that are (i.e. the network provider). Steering away of from these namespaces, and moving onto those controlled by content providers (e.g. URLs, and partially DNS), presents a different set of privacy concerns.

Even assuming that service providers are not concerned with user privacy, since privacy disclosure is transitive [42], user behaviour may disclose service information. The techniques described in Section 3.5.2 are usually applied to exploit user privacy, and mitigated as such. But they are equally applicable in determining which users visited which services or web sites i.e. they allow a competing service to build a costumer list for the competing services and products. As such while users might not be aware of these issues, services should be highly motivated to hinder this type of disclosure.

To assess this problem, Section 5.2 studies this issue within the context of location disclosure through the use of HTTP in the XMPP protocol. This is done trough a study of the of HTTP URLs that lead to location disclosure, either in the core protocol, in third-party extensions or due to implementation vulnerabilities.

To address this problem, Section 5.3 introduces techniques for service providers to conceal information in the multiple components of a URL at the application layer. This is generalised as an implementation for HTTP applications through the creation of a Session Bound Namespace (SBN), a transient URL namespace in which all names are URLs generated from other URLs, with varying levels of privacy as defined by a service policy. Similar mechanisms, for the network layer, are introduced in Section 5.4.

5.2 Informatin leakage at the application layer

Looking at the application layer it may be hard to grasp the status of privacy, for a given application or service. As pointed out (in Section 3.5.2) confidentiality is not the main concern, as soon as mechanisms such as TLS are in place. Side channels that disclose private information are the main practical concern in this context.

To tackle this, a study of location privacy in the XMPP protocol was conducted in [43]. It surveys a group of attack strategies and vulnerabilities that take advantage of URL handling in this protocol or through associated applications and services. The

primary goal of this work was to survey vulnerability to location privacy attacks using URLs, due to the use of HTTP as result of XMPP events.

The reason for choosing XMPP as a subject of study was threefold. First, because historically Instant Messaging (IM) services are considered to have a strong privacy model with regards to location privacy between users, messages always go through a central server to reach the destination. By comparison, the privacy model for web applications is weak, since each HTTP request reveals the user's IP address to the remote party.

Second the integration of this protocol with web services was widespread at the time. When [43] was published, both Facebook and Google supported this protocol for their respective IM services[1], which opens the door for door to leveraging web browser privacy attacks over this protocol and studying service provider practices in this area. In particular, the Facebook and Google services connect users to their social network rosters, which could open an attack vector for friends, co-workers or other types of contacts depending on the social network. While this protocol has lost favor in this context it is still common for internal communication, or as part of other protocol (e.g. IoT).

Finally, it is an area where protocol extensions are common practice. The protocol is extensible and client side implementations often take advantage of this to introduce capabilities that go well beyond the intended feature set of the protocol. Likewise since it is a federated protocol, different services apply different policies, to the point where feature implementation may differ with each service provider. Multiple extensions to the core protocol are available and published as a XMPP Extension Protocol (XEP).

5.2.1 Attack strategies

The underlying attach strategy, is to take advantage of the fact that some XMPP clients partially behave as HTTP clients and support a subset of their capabilities. In theory, if a client to client message could trigger an HTTP request to a resource controlled by the attacker (Figure 5.1), then this would be enough to determine user location based on IP address geolocation[20, 22].

Traditionally, to trigger this type of information disclosure attack in an IM protocol, the first approach is to trigger any feature that requires direct point-to-point communication between clients. File transfer is usually the classical vector for these attacks, but IM protocols usually prevent this information from being released without at least requesting consent from the target user. In this regard XMPP is very conservative: initially it only provided transfers going trough the server (in-band), and out of band exchanges require confirmation from the receiving user before they release information.

[1] Facebook removed XMPP support in 2015, and Google in 2017.

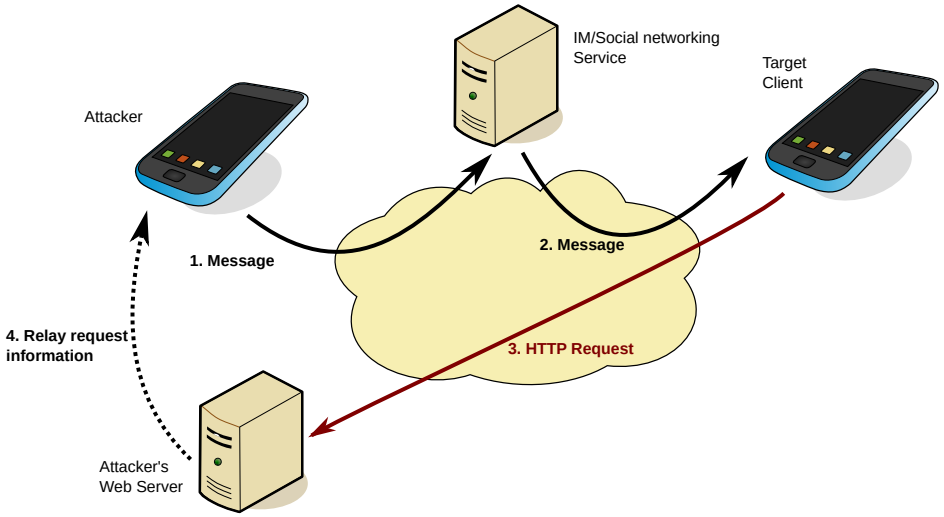


Figure 5.1: XMPP information disclosure attack via HTTP

As such, the primary target of work are attacks that do not require user interaction to succeed, and if possible, the target should not realise the attack is taking place. In particular three classes of vulnerabilities are of interest, in order of decreased generality:

1. Vulnerabilities in the XMPP protocol that can implicitly trigger HTTP requests.
2. Implementation practices that, while following protocol specification, unintentionally trigger these attacks.
3. Third party extensions that enable this kind of behaviour.

Some of these attack techniques are specific to XMPP while others can be reused with other protocols or applications that apply similar practices.

This work, illustrates the weaknesses that can be derived from three types of features: the use HTML content within messages, the retrieval of roster metadata using the HTTP protocol, and finally the use of client side extensions that automatically fetch data based on URLs found in messages. Of course, other attacks can explore different aspects of these weaknesses, but the work here should be enough to illustrate the privacy concerns that easily appear.

5.2.1.1 HTML Formatted messages

Protocol extension XEP-0071[193] introduces the use of HTML as message data format for XMPP messages. The original goal was to support text formatting with font sizes, colours and the inclusion of references to external data to be loaded using HTTP, in particular images.

The protocol draft adequately discusses the privacy implications of introducing HTML rendering within the IM client and provides recommendations to mitigate them,

from policies that delay HTTP requests until the user clicks the content to outright disable any form of data formatting and loading.

This method requires clients to support the protocol extension XEP-0071[193], with no requirements whatsoever on the XMPP server. An attacker crafts a message with links (either a text anchor or image) under his control. Since messages are sent to specific users in the roster, different messages can be sent to different users, establishing a unambiguous relation between each target and the links being retrieved.

While sending a direct message may prompt the target user to realise something is happening, HTML formatting also provides ways to conceal content. Images can be 1 pixel wide transparent images that look invisible, and the text description for URL links can be empty or contain invisible characters. The combination of these techniques, means the target might associate any type of action notification to a spurious error on the application rather than an action from the attacker.

5.2.1.2 Cross referencing HTTP and XMPP

XMPP clients exchange various metadata about the user. In particular, the protocol provides three extensions to exchange user icons (Avatars), XEP-0084[194], XEP-0153[195] and XEP-0054[196]. The three extensions differ in how this data is exchanged, but one feature is common among the three: all support the retrieval of avatars using HTTP.

XEP-0084 and XEP-0153 have additional requirements. Attacks targeting XEP-0153 require client support for external URLs in the vCard format, while attacks to XEP-0084 require server-side support.

The primary difficulty of targeting this feature is that this data is sent equally to all contacts in the roster. As such there is no way to target an individual user with this approach, because the server will forward the attack URL to all users. To circumvent this issue, this attack can be combined with browser fingerprinting techniques [197, 198] in order to match eventual HTTP requests with each user in the roster.

In this regard, XMPP clients behave just like web browsers: They advertise application specific information (application brand name, version, supported capabilities and operating system) among users (i.e. in band through the XMPP server). Likewise HTTP client implementations also send HTTP headers that expose similar information. This means browser fingerprinting techniques can be used from XMPP protocol information. Furthermore cross-fingerprinting between protocols seems to be a plausible way to overcome the main issue with this attack method, allowing us to identify an HTTP request as belonging to a specific contact in the roster.

Even if cross referencing fingerprints between protocols is not sufficient to get an unambiguous match, it is still possible to determine the correct HTTP fingerprint

from one of the other attacks to determine which contacts match the stored HTTP fingerprint.

5.2.1.3 URL preloading extensions

A final approach is to target client side application extensions that fall outside the scope of the protocol specification. Several client extensions automatically load URLs sent inside messages, in order to provide in-line previews for images and videos, or web page metadata. In practice, this is similar to the first attack method that was described using HTML data. The main difference however, is that such mechanisms are protocol agnostic, or are third-party components installed by the user. Earlier similar issues were described, for unintended data leakage through DNS, for different types of applications, in DNS hostnames in browsers [105] and in general URLs in other applications [106].

While this type of extensions will automatically trigger an HTTP request in the client, they also prompt user attention for the new messages. Unlike in the first case, there is no way to conceal the interactions, since extensions require visible text for the URL in the messages.

5.2.2 Results

There is a large number of XMPP client implementations and services, each with distinct policies, and evolving in time. This study targeted a set of representative client implementations and service providers that exhibit diverse behaviour that illustrates the discussion of this topic: it is easy to breach privacy in current network environments.

On the client side, 16 XMPP clients were surveyed. From this set, 8 are desktop applications, 3 run as part of web interfaces in a web browser, and 5 in mobile devices. The analysed desktop implementations were the following: Kopete, Gajim, Psi, Citron-IM, Pidgin, Messages, Digsby, Google-Talk and Jappix. For mobile devices, a selection of the top ranking clients in the application market were selected. In the Android operating system, Xabber, Beem and Jabiru. For iOS, Talkonaut and Trillian were used. Some popular clients are kept off the test set, to avoid clients based on the same implementation e.g. Adium for Mac is based on the same implementation as Pidgin and iChat client shares the same code base with the Messages client.

For the Google services, tests were conducted using both the web interface (through GMail) and the official desktop application for Windows. In Facebook services only the web interface was tested, since there was no officially supported XMPP client at the time.

	HTML Messages		HTTP Avatars		URL Preloading	
	Support	Vulnerable	Support	Vulnerable	Support	Vulnerable
Kopete	•		XEP-0153	•	•	•
Gajim	•	•	XEP-0153		•	•
Psi	•		XEP-0153 XEP-0084			
Citron-IM			XEP-0153			
Pidgin	•		XEP-0153			•
Messages	•		XEP-0153			
Digsby	•	•	XEP-0153	•		
Xabber			XEP-0153			
Beem/ Jabiru						
Facebook Chat						
Google Talk (Web)			XEP-0153			
Google Talk (Windows)			XEP-0153			
jappix.com	•	•	XEP-0153			
Talkonaut/ Trillian			XEP-0153			

Table 5.1: Results: XMPP Client vulnerability for each attack

Tests were conducted on the *jabber.org* or *jabber.cz* services, since they are know to support all the required features, and also on the Google and Facebook XMPP services when applicable.

Table 5.1 presents the summary of the results for each attack method in the various XMPP client implementations. From the set of 16 clients, 5 were compromised. The only attack method that succeeded against Pidgin, was through its URL preloading extensions, however this extension is disabled by default. The majority of web-based and mobile clients lack support for HTML formatted messages.

Additional findings about the multiple implementations and services for each of the aforementioned attack methods are discussed more thoroughly in the following subsections.

5.2.2.1 HTML formatted messages

From all tested client set, only 7 supported HTML formatted messages, and only 3 were found to be vulnerable to this method. However the HTML content was handled

differently by different implementations. The different client behaviour can be broken down into four distinct groups based on how this feature was implemented:

1. Kopete, Gajim and Digsby use the WebKit library[2] to render HTML content, meaning they actually run a browser engine within the XMPP client application.
2. Psi renders HTML using a Rich Text parser, thus preventing any form of remote request from the HTML parser.
3. Pidgin uses a regular text parser that allows some HTML tags. Messages for Mac follows a similar approach.
4. Jappix.com was the sole web interface that allowed HTML formatted messages.

All successful attack cases used image tags to trigger the HTTP requests. Other attack payloads, such as links, CSS stylesheet references, or the inclusion of Javascript in the HTML payload were found to be ineffective.

5.2.2.2 HTTP published avatars

On the client side, while support for avatar information in XMPP was possible through multiple extensions (XEP-0153 and XEP-0084), only the former seems to be widely supported by client implementations. From the 16 tested clients, only 2 (Digsby and Kopete) were found to be vulnerable, in both cases using XEP-0153.

On the server side, the Facebook chat does not support this protocol and any attempt to set an avatar, through one of the other extensions will cause the connection to the server to be terminated. Likewise Google Talk does not implement server side support for XEP-0084 at all.

Since the avatar information was sent to all contacts in the roster, the primary limitation of this attack was that it could not target a specific user. Previously it was assumed that browser fingerprinting techniques could be reused to assist in this task. This proved to be true. For both the cases that succeeded in triggering an HTTP request for the Avatar image, the clients used a User-Agent header in the request which clearly identifies the client application (Table 5.2). While Digsby only provides a terse application specific header, Kopete uses the regular header from the WebKit library which is not specific to the XMPP client.

[2]https://webkit.org/

Client	HTTP User Agent
Kopete(Avatar)	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/533.3 (KHTML, like Gecko) kopete/4.8.1 Safari/533.3
Gajim	Gajim 0.15
Digsby (HTML messages)	Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.10 (KHTML, like Gecko) Chrome/8.0.552.5 Safari/534.10
Digsby(Avatar)	Python-httplib2/digsby
Pidgin	Mozilla/5.0 (X11; U; Linux x86_64; en-us) AppleWebKit/534.16+ (KHTML, like Gecko) Version/5.0 Safari/534.16+

Table 5.2: HTTP User Agent used by each XMPP client

5.2.2.3 URL preloading extensions

Concerning URL preloading extensions, only 3 of the surveyed implementations supported this type of extensions, and when enabled all of them were found to be vulnerable (Pidgin, Gajim, Kopete). In particular they immediately triggered an HTTP request to fetch an image, even before the receiving user takes any action.

The only observable difference between these three implementations was that Pidgin did not use any particular revealing information in its user agent (Table 5.2), while both the other implementations clearly identify the XMPP implementation

5.2.3 Discussion

The purpose of this work was to experimentally understand how privacy leakage can work in practice. Lessons learned reveal different aspects: how widespread some privacy breaches are, some fundamentally weaknesses and if these could be seen as targets to be addressed.

Given the target test set, only a moderate number of implementations were found to be vulnerable to the attack designed her designed here. In general, both XMPP client implementations and services seem to be very conservative when it comes to adopting protocol extensions that would allow this type of privacy breach, in particular in mobile devices.

Even in services where embedded web content is common, the use of embedded HTML content was not supported (with the exception of jabbix). This privacy feature happens despite the fact that the web versions of these services do protect users from this type of privacy breaches. For example, if a Facebook user posts a link to an image (on the web site), the Facebook services fetch a copy of the image to display on their webpages which avoids disclosure of user location to a resource controlled by an attacker. Other services employ similar techniques, Twitter for example only displays content from pre-approved sources (Youtube for videos or established image hosting services for pictures) otherwise the user will only see a clickable URL.

Interestingly enough, these approaches are not implemented in the XMPP clients, which either disable the feature completely or are vulnerable to attack. This would be understandable for client implementations that do not provide their own hosting infrastructure, but even services like Google or Facebook that already implement this kind of privacy features in other contexts simply disable these features in XMPP.

Earlier the use of browser fingerprinting techniques was proposed to circumvent the main limitation for one type of attack. The initial expectation was that, when issuing an HTTP request, XMPP clients might be identifiable, much like browser users with a unique fingerprint [197]. Some clients clearly embed the XMPP client name as part of the HTTP User-Agent header. This is sufficient to identify the specific implementation, but it did not reveal additional information, and requests were indistinguishable between two users with the same implementation. Knowing the implementation may however enable more targetd privacy attacks.

It is not unusual for client implementations to reveal application information as embedded metadata. HTTP browsers and mail agents both have the User-Agent header, and XMPP advertises application name and operating system in band. What is novel here is consistent (in some cases) cross-protocol User-Agent information.

In HTTP a similar trend has emerged as established practice where consumers for HTTP APIs, where Terms of Service (ToS) mandate an application must use a distinguishable User-Agent header, even when more secure forms of identification are available. As of January 2018 similar obligations can be found in the requirements for third-party application using popular web service APIs like Reddit, Discord, Github or Travis CI. This suggests similar types of privacy information could be accidentally disclosed by these applications. In fact even if this test was dated in time, the trend towards automated data handling (e.g. IoT), suggest this problem has potential for growth.

5.3 Service provider enforced privacy for URLs

It is now established that privacy disclosure from names may be a consequence of multiple independent issues. Protocol design, implementation issues, third-party extensions and user-behaviour can result in privacy disclosure. In some namespaces are not user defined or controlled i.e. the namespace is partially or completely controlled by other entities, often a service provider that handles name assignment.

There is ample previous work that mitigates URL disclosure from side channels (Section 3.5.2) through client side implementations. HTTP browsers in particular are a common research target on this subject, but other protocols such as SMTP also exhibit similar issues and implementation workarounds.

Even outside the context of side channel attacks, it is worth considering the amount of metadata placed in a URL. URLs hold a reasonable amount of metadata about interactions going through the network, revealing which services where visited, or who the user is [174, 199, 200]. Either through remote exploitation or forensic analysis [201, 202] of stolen devices, a significant amount of information can be extracted from URLs alone.

On the service provider side there are examples of ad-hoc privacy oriented policies in general purpose protocols. For example, both Google and Facebook apply content specific policies in HTTP websites and XMPP endpoints, to prevent privacy disclosure between users. Similar policies are implemented by email providers that scrub revealing information from SMTP transactions.

It is important to remember here that, in the real world, privacy is a consequence of mutual enforcement within a space e.g. altough privacy is the result of individual discretion, on a day to day basis people hold different expectations of privacy from different locations. Some spaces are more private due to physical barriers, others due to the letter of law or agreed human behaviour. This real world notion of variable privacy does not translate into "online spaces". The technical burden of privacy is often considered to be a user problem, or part of specialised infrastructure (e.g. ToR[203]), and the service provider rarely takes action to protect user privacy.

A comparison can be drawn to cache oriented ICN network architectures (i.e. CCN, NDN), where a variation of this privacy problem is the ability to probe intermediate router caches for the presence of content. This remains an open problem even in these architectures [161].

To reduce private information exposure in URLs, this section introduces mechanisms for encoding URLs in a way that conceals private information, shifting the burden of privacy to the conent provider. For implementation purposes the work presented here is focused on encoding of URLs used in HTTP applications, since these have specific requirements regarding backwards compatibility and support for DNS hostnames. Based on a set of requirements a new namespace (a subset of URL) is defined that conceals information in each URL component. The example namespace defined here is meant as a transient URL namespace, used with distinct user sessions, i.e. a Session Bound Namespace (SBN). However the notion of session could be redefined based on service provider policy to have different meanings, such as a time limited namespace, or a distinct URL namespace based on user location. Overall this mimicks the different privacy properties of spaces of physical in reality.

5.3.1 Requirements

URLs are used in various network protocols to identify and locate related resources, such as content, users, or the network attachment point for a service. These names are assigned based on parameters such as resource availability, human recognition, brand value or semantic meaning. While properties, such as resource availability, are directly related to the network resources or data from the service provider, others may be defined arbitrarily.

The main challenge to be overcome is to enable a service provider to, at will, start providing service under a transient URL namespace, that can be converted back to the original canonical URL namespace. Furthermore, as a practical constraint it is desirable that this can be implemented strictly as a server component. Both clients and other external observers in the network will see URLs belonging to that transient namespace instead.

The mapping function that converts regular URLs into this namespace is designated E_{SBN}

$$E_{SBN}(URL) \rightarrow URL'$$

and conversely the reverse mapping function is D_{SBN}

$$D_{SBN}(URL') \rightarrow URL$$

In an HTTP interchange (Figure 5.2) D_{SBN} decodes incoming URLs values that are in the namespace, while URL links sent to the client in the response are encoded with E_{SBN}. Regular URLs can still be used to reach the service, e.g. for the initial contact, but the service provider initiates a new session bound namespace by redirecting the browser to the correct URL (which may also be co-located with the server for the original URL).

Formally the goal is to map a namespace of all URLs used by a service into multiple session specific URL namespaces which are more suitable for privacy purposes. The three key properties of this new namespace are the following:

1. Transient: URLs in this namespace are only useful within the session that generated it, and any use outside that session will result in an error.
2. Security: unintended parties should not be able to trivially reverse the mapping.
3. Extensible: URLs can hold ancillary information, placed therein as part of the conversion process. This may be required to uphold the previous two points.

Section 3.3.5 already described the multiple components that constitute a URL and internal limitations of each component. However, before looking at how URLs are

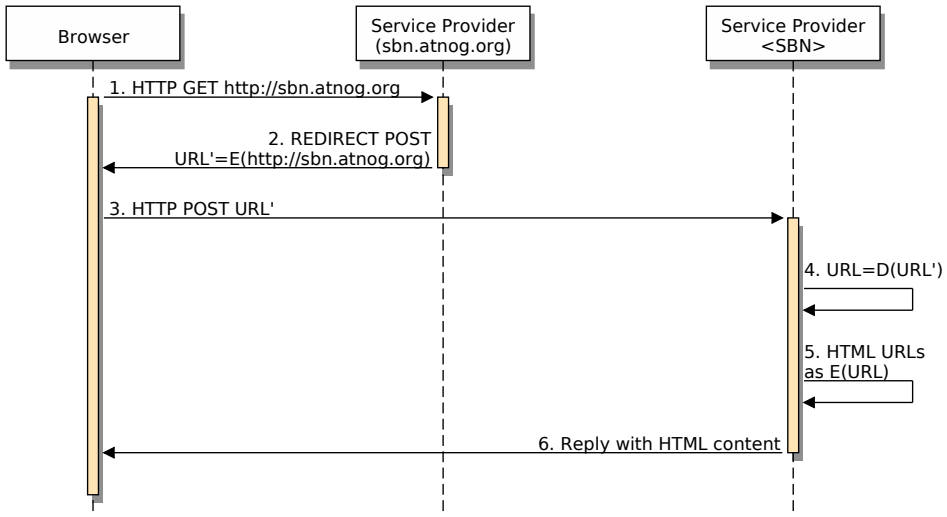


Figure 5.2: SBN implemented at the server side

assigned in the Session Bound Namespace (SBN), some additional constraints need to be introduced for the purposes of keeping compatibility with HTTP applications. Only then can a valid mapping function be defined.

5.3.1.1 HTTP Backward compatibility

To retain backwards compatibility with HTTP in a way that minimises the required changes to existing code base, some initial requirements are defined for the functions E_{SBN} and D_{SBN}. These are meant as a compromise between the intended privacy goals and current practices. Later these will be discussed again in light of practical results.

From the previous breakdown of URL components (Section 3.3.5) and empirical observation in existing HTTP applications, a group of six requirements can be defined:

1. The *Scheme* defines the protocol handling at the client, and therefore must not be changed. Otherwise the client implementation would not recognise the URL scheme.
2. The *Username* and *Password* in the *Authority* segment, as seen in HTTP browsers, is left unchanged. In practice, browsers discourage the use of inline credentials in URLs.
3. URLs in the same session must not break the same origin policy, i.e. they must use hostnames under a common domain. Otherwise, browser features like cookies or cross domain security checks will prevent applications from working. However different sessions can use distinct hostnames.
4. Protocols (e.g. HTTP, FTP) navigate the URL *Path* using relative references. As a consequence, relative references must also work for URLs in the SBN - thus the number of segments in a mapped path must remain unchanged.

5. For some HTTP based applications, the URL *Query* is manipulated at the client side. Ideally the mapped query attribute names should not clash with the ones used by the application. A less common case can be seen in JavaScript applications, where the client expects to read the *Query* string; in such cases the mapping function should not alter it.
6. The *Fragment* is considered to be a local (client-side) reference, and can point to protocol specific content. Moreover the URL fragment is typically not sent as part of a request (HTTP), so there is no gain in modifying it for privacy purposes. If the concern is that these expose content information, they can be replaced with hashes, but this is not covered in this implementation.

Paths are frequently manipulated at the client side, using relative links and known path segment names. In order to support compatibility with these cases, an additional requirement is introduced for the *Path* component:

7. A path can include a mixture of segments encoded and non-encoded according to the SBN.

Requirement 7 in particular is a compromise that relaxes the goal of privacy for compatibility. It facilitates the adoption of this scheme where it would otherwise require generating custom per session Javascript code.

Two conclusions about the desired function can be extracted from requirements 4 and 7. First, one must be able to distinguish encoded and non encoded *Path* segments. And second, path segment transformations must be independent from one another i.e.

$$E_{SBN}(Path) = /E_{SBN}(Path_0)/E_{SBN}(Path_1)/...$$

$$D_{SBN}(E_{SBN}(Path_i)) = Path_i$$

And naturally it follows that using the reverse mapping function in a *Path* that was not encoded, returns the same *Path*.

$$D_{SBN}(Path_i) = Path_i$$

For purposes other than web based applications, this set of requirements may be unnecessary. But for now all these assumption are left in place. Later Section 5.3.4 and Section 5.3.5 will look into their consequences and strategies for relaxing them.

5.3.1.2 Viable mapping functions

Since the $E_{SBN}()$ function needs to be reversible for the service provider to be able to determine the original URL, then the service provider either stores mappings between

all URLs, or the new URL' contains all the original information encoded within their new representation.

The first option is typically used by URL shortening services: they generate a unique short URL that maps into the original URL, and then lookup the new URL. However this approach does not fit well into this type of scenario. For a distributed service provider it would require a synchronised method to generate new URLs, and a distributed mechanism to resolve them. Furthermore since the namespace is transient, multiple SBNs can be served at the same time, e.g. one for each user of the service, which further expands the number of stored mappings

A second option is to encode the original URL information as part of the new one. To prevent third-parties from reversing the transformation this means encrypting the content before encoding it as part of the new URL. This is the approach followed here.

Choosing an encryption scheme for this task requires determining the best compromise between the overhead and limitations of longer URLs with the performance costs of different encryption schemes and deployment choices. Nevertheless the first step is to cross reference the properties from the URL namespace with the requirements defined earlier, to define this new namespace.

5.3.2 Session Bound Namespaces

Based on the requirements identified in Section 5.3.1 the mapping functions can now be defined. When converting an URL into the SBN, the original URL components are encrypted (designated as $K()$, where K is one of the encryption schemes that were studied). The choice of encryption schemes presented here is not meant to be exhaustive but illustrative: it covers some of the more popular off-the-shelf implementations schemes available for different types of encryption.

For symmetric encryption, two schemes are used. AES-CBC-128[3] and Salsa20[4]. The later is a stream cipher, and requires the caller to ensure non repeatable nonces are used with the same key. For asymmetric encryption both RSA, and Elliptic Curve Cryptography[5] are considered.

The encoding scheme for each URL component is described separately, namely the *Authority*, *Path* and *Query*. The remaining components *Username*, *Password*, *Port* and *Fragment* are not described since they are left unaltered for compatibility purposes. Table 5.3 shows an example of the output of the E_{SBN} function for each individual component.

[3] AES-CBC-128 as used by the *python-cryptography* module, with PKCS7 padding and HMAC256
[4] As used by the sodium/NaCl implementation
[5] Standard ECC curves in the *seccure* implementation

Table 5.3: Encoding examples for different URL components ($K()$ is ECC/p256)

Component	*Input*	$E_{SBN}(percomponent)$
Hostname	sbn.atnog.org	ad4cbfteczl5i5jlqdyop7ji7lb7kxsy3lgcvqzcw5i5ks3kt4kdsr exsc5deqw.s7moiamauu7uogxhmi53dwj5b3m y36bh2c3wkkqgfi7rf3eydqwuq.sbndomain.tk
Path	/hello	/@AUNuJ4C2jyxQxkWdN- jKX689p4wLfInywtEqTiAVW 4EgazeDo4Wq1n84iXJm2JjL0A
Query	?action=42	?sbnquery=AL8Z-RdNVJ5-41- Oqu3K09l4xhOy8mmorBz j3xnDNXVNI2PPjVje2cHtC0LMH3dlI8J7fw

5.3.2.1 Authority

For this component the hostname must be replaced with a new one, which is apparently unrelated to the original and unique for each session. For example $E_{SBN}(sbn.atnog.org) \rightarrow session1.sbndomain.tk$ are to an observer unrelated host names.

This means the new hostname should not be based on the original domain name, e.g. a subdomain. But hosts for different sessions can be part of a common domain (e.g. session1.sbndomain.tk and session2.sbndomain.tk). This problem can be subdivided into the following points:

1. Efficient allocation and management of bulk DNS names as session identifiers that point to the correct servers

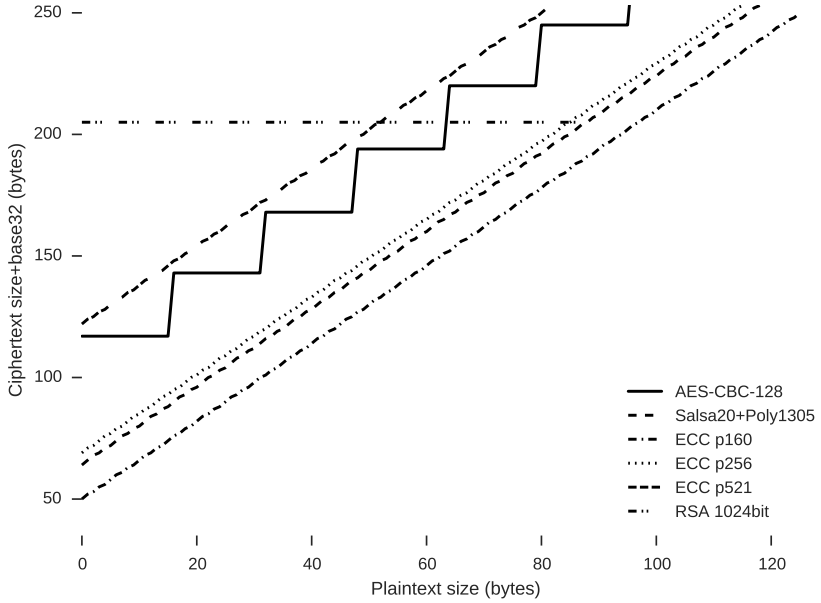


Figure 5.3: Host overhead for multiple encryption schemes

2. Verifying a hostname as being valid for a given SBN

As such the function must allocate a new DNS hostname for the session. Even assuming this process could be time consuming, domain names can be acquired and configured in bulk. The process of configuring one hostname to point to the proper host would take at most a DNS update [204], provided the TTL for the hostname is low. If necessary, the DNS hosts can be preconfigured, long before they are needed by the service provider and used as needed.

Alternatively, each individual hostname doesn't have to actually exist as a single DNS record. A wildcard rule can be enforced at the authoritative DNS server for a subdomain to the same set of servers.

The hostname part of the URL is converted as follow:

$$Host' = base32(K(Host)).TLD$$

Where the *TLD* is a separate level domain under the control of the service provider, and *Host* is the original hostname. Likewise, reverse mapping function would decrypt the hostname to extract the original hostname.

Figure 5.3 shows a comparison of the overhead for different encryption schemes (K). The total sizes include the overhead for base32 encoding. Even using the most expensive scheme (ECC/p521 in Figure 5.3) would still allow us to encode 80 bytes within the new hostname. Using a 20 byte nounce, implies the original hostname must not exceed 60 bytes. RSA is always padded to a constant size, and a larger RSA key cannot be used because the ciphertext would be larger than 250 bytes. The 1024 key size implies a plaintext larger than 86 bytes cannot be encrypted (maximum plaintext size when using RSA-OAEP).

For the example, in Table 5.3, since the encoded content is larger than 63 bytes (the maximum size for a DNS hostname label), it was split over two labels.

5.3.2.2 Path

As outlined previously in Section 5.3.1.1, for compatibility purposes each path segment is encoded separately in order to retain the original *Path* structure.

If *K()* is an encryption function then an individual path segment will be encoded as $@base64(K(segment))$. The full path is encoded, one segment at time, as

$$/p_0/p_1/... \rightarrow /@base64(K(p_0))/@base64(K(p_1))/...$$

The "at" symbol(@) is used here as a special marker to distinguish between *Path* segments encoded into the namespace, and those that are not (Section 5.3.1, *Require-*

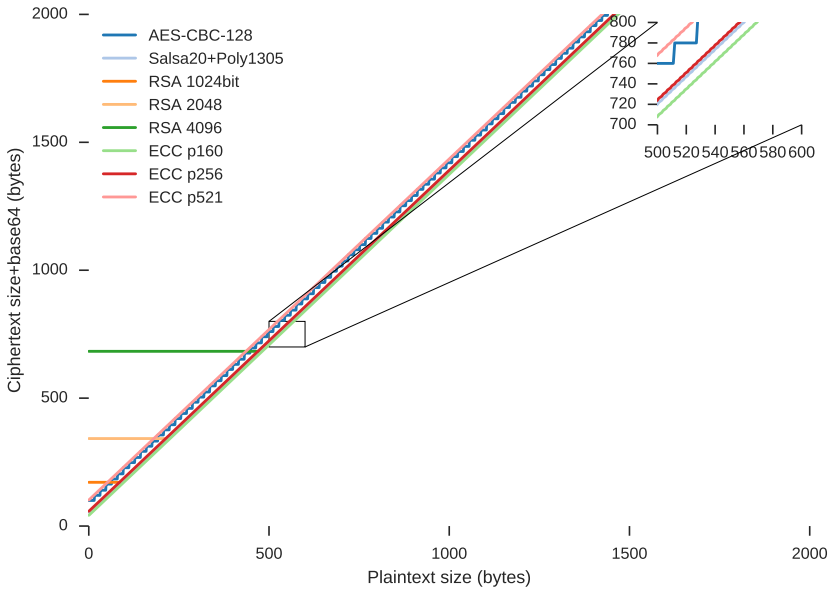


Figure 5.4: Path segment overhead for multiple encryption schemes

ments 4 and 7). This marker must not appear in non-SBN URLs, which led to choice of @, but other path marker rules may used instead.

The encoding overhead for each path segment for different encryption schemes can be observed in Figure 5.4. As in the previous case, RSA (PKCS7) is also included, now for larger key sizes to allow for larger payloads, but as before they are limited in the maximum payload (RSA1024 - *86 bytes*; RSA2048 - *214 bytes* ; RSA4096 - *470 bytes*) and padded to a constant size.

5.3.2.3 Query

Query arguments can be concealed by encoding the original query inside a new query argument, using a reserved query attribute name for this purpose. For example using *sbnquery* as the attribute name:

$$Query = sbnquery = base64(K(Query))$$

Data encoding is identical to the *Path* segment case, the encrypted query is encoded as base64 and included as the value for the *sbnquery* attribute in the new query.

5.3.2.4 Additional Considerations

The previous definitions suffice to setup a Session Bound Namespace (SBN). Before proceeding to discuss how to implement it, some additional considerations can be made concerning this namespace.

While the previous scheme conceals information through encryption it does not necessarily enforce uniqueness for different sessions. The underlying encryption scheme

may guarantee that encrypting the same plaintext twice will result in different ciphertexts, but if this assumption does not hold two strategies can be considered:

1. Use distinct keys for each session, and identify keys based on signed HTTP cookies, or similar authentication strategies for HTTP.
2. If using a single key, insert a random session specific nonce (e.g. 20 byte UUID) as part of encrypted payloads.

Relying on encryption for the various URL components implies additional overhead when handling these URLs. From the encoding methods described in this section, processing a URL would require a number of encoding operations that increases linearly with the number of path segments as

$$K(hostname) + \sum_{i=0}^{N} K(Path_i) + K(Query)$$

For a detailed performance evaluation, it is best to consult with detailed benchmarks for each implementation and performance may vary as software[6] and hardware implementations of these schemes become available. Another way to amortise the cost of these operations (at the expense of memory) is through caching of encryption/decryption operations. This particular approach will be covered further in Section 5.3.4.2.

5.3.3 Implementation

After surveying the namespace limitations for the creation of a Session Bound Namespace (SBN), the assignment and binding functions for this namespace can be implemented for a specific protocol. For implementation purposes, HTTP was selected, since it is the one of the most widespread protocols and it can be implemented strictly at server side without requiring changes to client implementations. In particular in the context of web based applications.

An instance of this implementation is available for consultation at *http://sbn.atnog.org*, and its source code is also publicly available as an Open Source package[7].

Let us consider an example of how an SBN is bootstrapped when using an HTTP service (Figure 5.5). As is common is HTTP exchanges, the user starts by browsing using a well known URL for a service (in this example *sbn.atnog.org*). The Service Provider (SP), based on internal policy, decides this request should use URLs in the Session Bound Namespace (SBN) and responds with an HTTP redirect (step 3 in

[6] http://www.cryptopp.com/benchmarks.html
[7] https://github.com/ATNoG/flask-sbn

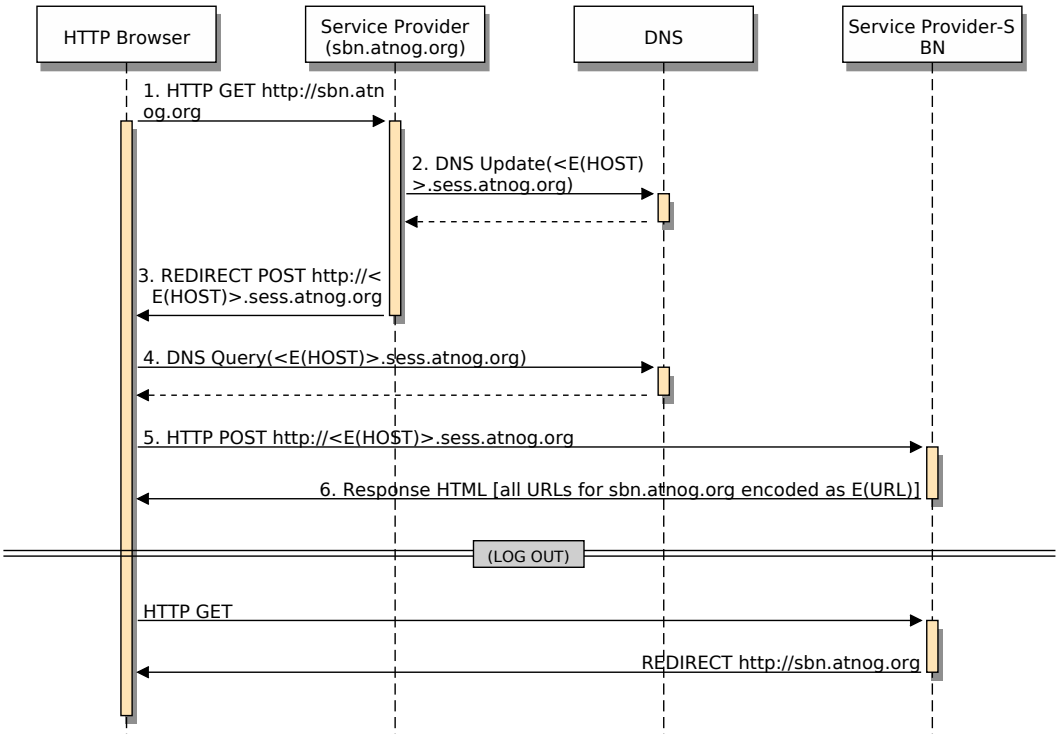


Figure 5.5: SBN workflow for an HTTP browser

Figure 5.5). This decision can be made based on specific policies, e.g. if the user is about to login into the website.

For the HTTP case in particular, some care must be taken during the redirection step: since the hostname in the URL will change, cookies established earlier will not be in effect due to the Same Origin Policy (SOP). The original Service Provider instance should redirect the browser along with a signed token to establish any required session information at the new instance, as is the case in any HTTP login procedures that crosses over different domains,

For this instantiation, step *2* in Figure 5.5 is skipped. Instead a wildcard DNS configuration is defined at the authoritative DNS server, as pointed out previously in Section 5.3.2.1. This is only possible because in this instance all sessions use the same TLD.

A relevant observation for implementation purposes is that many web development frameworks already apply dynamic control over URL generation for links within an HTML page that point to the page hostname. Popular web frameworks such as Django, Ruby on Rails, or JSP, generate HTML content based on page templates, where internal URLs are formed automatically based on dynamic settings (service hostname, path locations or localisation settings). In practice, this matches well with the previous requirements, and any such framework should be well suited for this purpose, since they already implement the logic required to handle dynamic URL schemes, due to

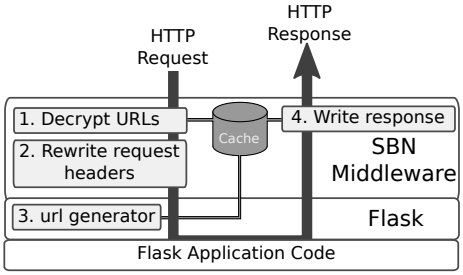


Figure 5.6: Functional diagram for SBN implementation

virtual hostnames, component path discovery and contextual URL creation.

In the HTTP privacy solution described in [173], their goals are achieved through the implementation of a transparent HTTP proxies that rewrite HTML content. Instead it is preferable to implement this logic as close as possible to the service provider web framework, because the previous observation means that is where the actual web site URLs are assigned and there is room for further optimisations. For testing and instrumentation purposes, this thesis actually implemented both approaches. The first is ideal because ties SBN policy to the application, but the second provides the tools used for experimental evaluation with existing websites.

The implementation described here is based on the *Flask*[8] web development framework, and is composed by the following components: (Figure 5.6): *1)* a middleware that handles incoming HTTP requests, decrypts URLs in HTTP headers; and *2)* a middleware replaces them with the decrypted values; *3)* a set of URLs generator stubs, that replace the ones used by default in application code or as part of the template engine and rewrite all URLs created in HTML content (encrypting URLs in the SBN); *4)* a module to rewrite HTTP responses.

All components share the same SBN configuration parameters that define how URLs are generated (encoding and encryption schemes). Communication between components happens either through a shared context associated with each HTTP request or through a shared cache. The later is used extensively to store encryption and decryption results for the same request, and for a single request identical URL component encryption/decryption operations are retrieved from this cache.

By default, the application code is unaware of the real URLs being used, as it always uses the original decrypted URLs. This is only possible because internally application code in Flask (as is the case in other frameworks) rarely handles URLs directly, instead it builds URLs based on internal APIs that go through the URL generators building them one component at a time.

Mechanisms for the application logic to take over URL transformation are also

[8]http://flask.pocoo.org/

implemented. This is required not only for the redirection example described earlier in Figure 5.5, but also in scenarios where application logic wants to enable/disable the of SBN URLs based on a fine grained policy.

5.3.4 Results

To evaluate the impact of this solution, the HTTP traffic for a group of popular websites was analysed, based on the top websites in the Alexa[9] ranking. Some websites from that ranking are not included, since they are optimised to display minimal content on load, or defer content loading using Javascript, making it harder to perform automated analysis.

The evaluation methodology consists on instantiating a transparent HTTP proxy that performs URL transformation based on various SBN policies, and accessing the target sites through this proxy. All requests between the HTTP client and the proxy operate as depicted in Figure 5.5, but instead of fetching content from a local database or disk, requests are made to the target websites and the corresponding responses are then rewritten to enforce the Session Bound Namespace (SBN).

This type of approach will always incur in a significant delay when compared with not using the proxy. However the primary goal is to analyse the impact caused by these methods with relation to the amount and type of URLs on a website, such as URL length (number of path segments due to *Requirement 7* from Section 5.3.1). Real world websites provide results on the actual practices being used composing URLs in websites. Throughout this analysis, the primary goal is to determine the number of encryption and decryption operations required to enable the use of SBN in these websites and the potential delay of the these operations.

As a starting point for this analysis, the most strict scenario is considered when all links in a web page are transformed according to the Session Bound Namespace (SBN), including links and references to other websites (i.e. other domains). The following subsections will expand on these preliminary results through the implementation of different URL transformation policies that relax the requirements identified earlier and analyse how they affect these results.

When an HTTP request arrives at the proxy, all URLs in headers that belong to the SBN are decrypted. Conversely when handling an HTTP response, all URLs in the headers and the HTML content (anchors, images, css links, etc) are encrypted. As such the impact of this approach depends on the number of HTTP requests required to load the page, and the amount of URLs included as part of HTML/CSS content.

Different web pages exhibit distinct load behaviours concerning the number of HTTP requests (Figure 5.7) and HTML URLs (Figure 5.8) that must be encoded

[9]https://www.alexa.com/topsites (2015)

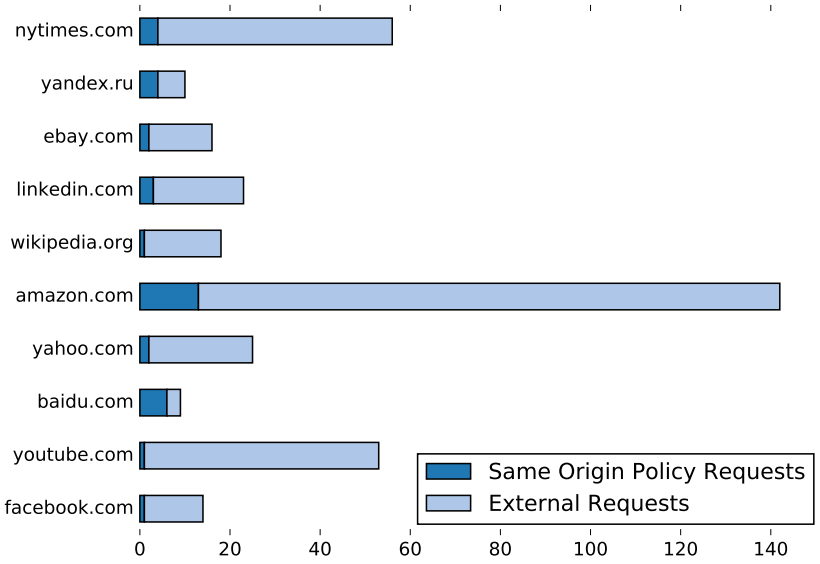


Figure 5.7: Number of HTTP requests

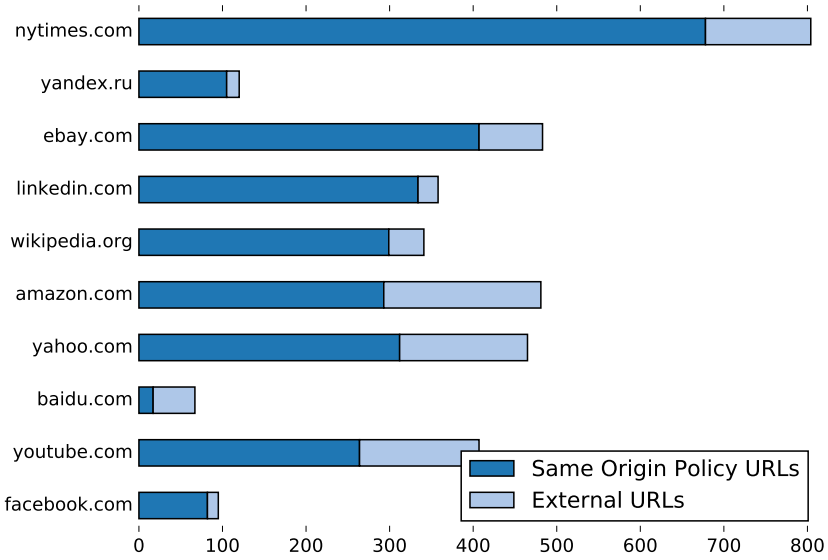


Figure 5.8: Number of URLs in HTTP content

according to the SBN. Both these figures distinguish between URLs that fall under the Same Origin Policy (SOP) as the visited website (i.e. use the same domain/port). This distinction is not relevant yet since all URLs will be encoded/decoded according to the SBN, but in later optimisations these values will be revisited.

For reference, an existing survey on this topic[10] concludes the average web page contains 100 objects references. By comparison the values seen in this test set are much higher. The highest value is seen for the *nytimes.com* web site (Figure 5.8), but some of the other web sites engage in content loading that defer HTTP requests that rewrite the page until the user performs some action (e.g. scroll) instead of loading it

[10]http://www.websiteoptimization.com/speed/tweak/average-web-page/ (2014)

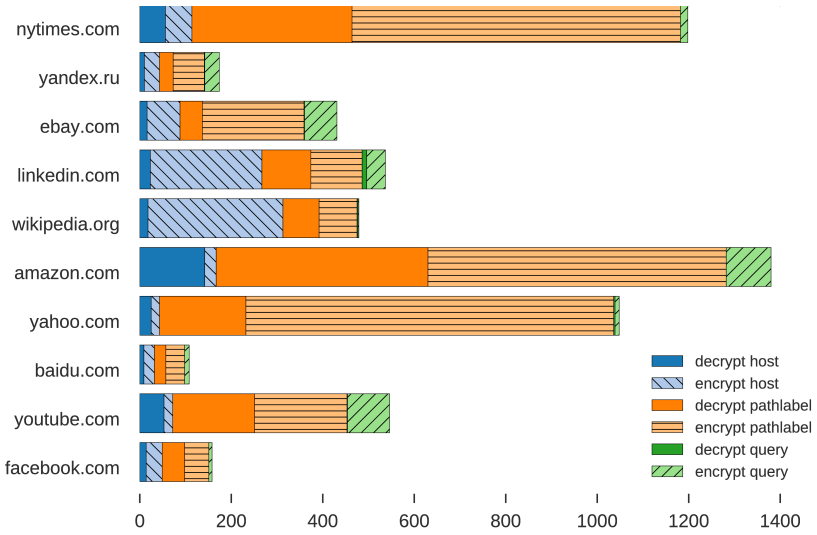


Figure 5.9: Encryption/Decryption operations for different web pages

all at once, which may explain this discrepancy. For the presented measurements, the amount of HTTP requests includes all requests since the page starts loading, until it is fully visible in the browser, and the browser stops issuing new requests for content.

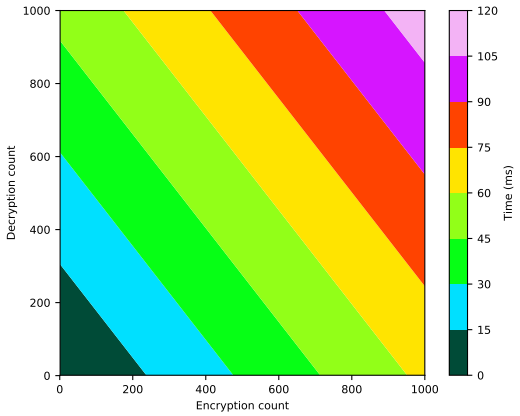
Figure 5.9 shows the amount of decryption and encryption operations (per URL component) that were required to serve each of the analysed web pages. Web page values are broken down by encryption/decryption operation and the URL component being transformed (host, pathlabel, query). The amount of required operations is certainly high, which is not surprising since it includes all requests involved in fetching webpage (i.e. images, CSS, javascript, HTML, etc), and in some cases this involves a large number of HTTP requests (Figure 5.7), e.g. amazon.com (142 requests).

For the purposes of estimating how the previous results can impact request handling delay, a microbenchmark for encryption/decryption operations is included in Table 5.4. Presented times (in milliseconds) are averages of 3000 executions performed on a Intel i5-4570R CPU at 2.70GHz. The remaining ECC curves and RSA are not included, since their execution times are always higher than the remaining schemes and RSA is in general too expensive for these purposes.

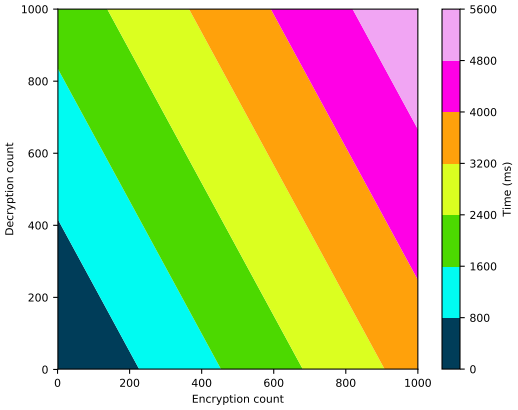
Table 5.4: Encryption/decryption microbenchmarks for a 1000 byte plaintext (average per operation)

Scheme	Encryption(ms)	Decryption(ms)
Salsa20+Poly1305	0.063	0.049
ECC 160	3.524	1.914
AES-CBC	0.235	0.238

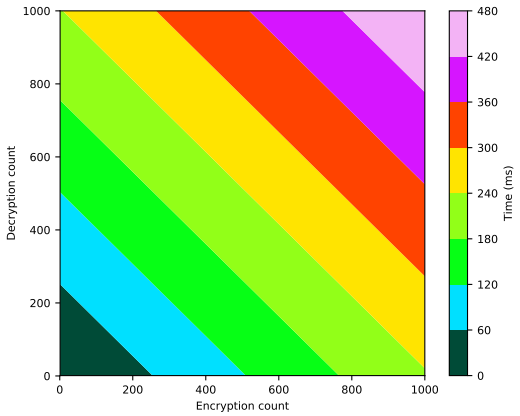
From the initial scenario, considering the *amazon.com* case from Figure 5.9 (with 776 encryption and 605 decryption operations), would require a total delay of *79 ms*



(a) Salsa20+Poly1305



(b) ECC 160



(c) AES-CBC

Figure 5.10: Estimated total Encryption+Decryption delay based on op-count

(Salsa20), *326 ms* (AES) and *3892 ms* (ECC).

These estimates can expanded for variable amounts of encryption and decryption operations to highlight the desired target delay zones. Figure 5.10 plots the time delay (as a color bar) for a number of encryption or decryption operations. For algorithms where the is asymetric (e.g. decryption is more time consuming than encryption) a slope can clearly be seen. This provides a way to quickly estimate the encryption/decryption delay of SBN based on an upper limit on the amount of operations that can be performed when serving a page.

Figure 5.11 details the data used to generate table Table 5.4, and shows the 99.999% confidence interval for encryption and decryption times as the payload size increases. It can be seen that the plaintext size holds little impact in the delay for each operation. While AES has similar times for encryption and decryption operations, in the other schemes encryption is more expensive than decryption. This is the worst case for this data set, since encryption is more common than decryption. The same relation can be observed as the slope in Figure 5.10.

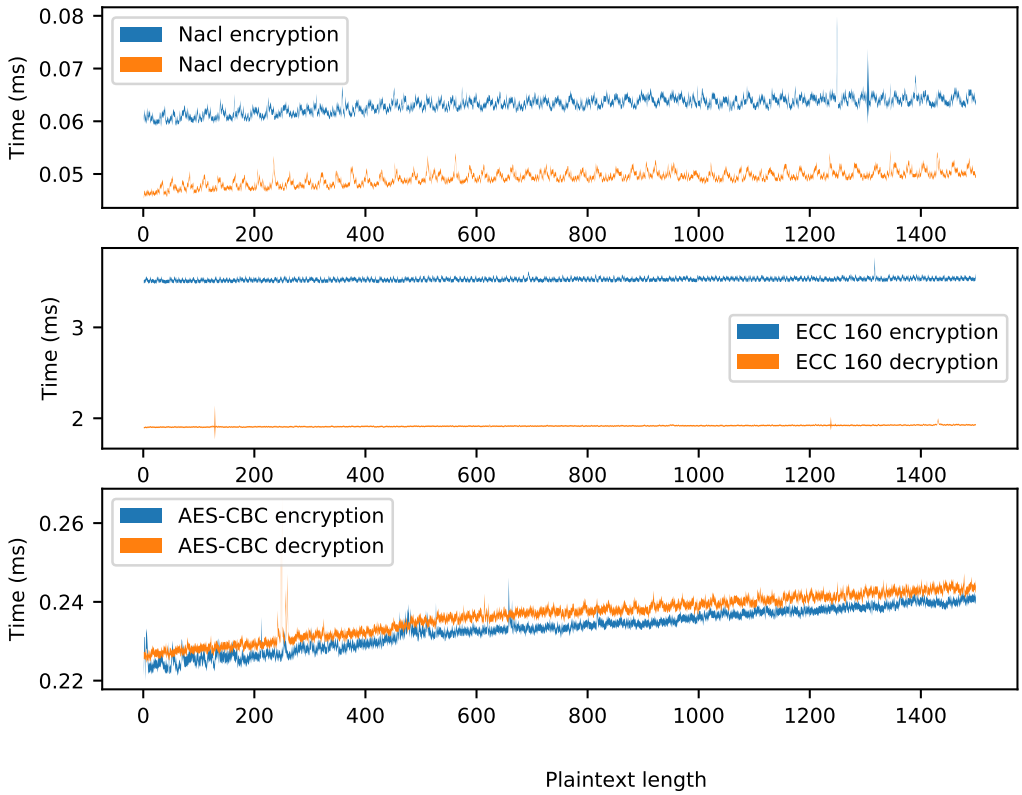


Figure 5.11: Encryption/Decryption times vs plaintext size.

As it stands the current delay for serving the heaviest page on the test set (*amazon.com*) with the most expensive encryption scheme seems unacceptable, at nearly 4*s* just for encryption/decryption of URLs. The remaining subsections will introduce different optimisations, through various policies implemented by the service provider with regards to which URLs should be privacy protected. First, by restricting the Session Bound Namespace (SBN) to URLs that fall under the Same Origin Policy (SOP); second, through the introduction of widespread caching of encryption and decryption operations across multiple requests; and finally, through the encoding of URL paths in a single operation.

5.3.4.1 Restrict SBN to Same Origin Policy content

Since the goal of this implementation is to introduce a privacy preserving namespace for content served by a specific service provider, a policy more aligned with these privacy goals would be to only encode URLs that fall into the Same Origin Policy (SOP) as the main website. That means, URLs under a different domain would not be encoded, and would be left unchanged or served as without changes. In particular, for the studied group of web sites, this avoids encoding URLs in HTTP requests from Content Distribution Network (CDN) hosts, or encoding links pointing to external domains.

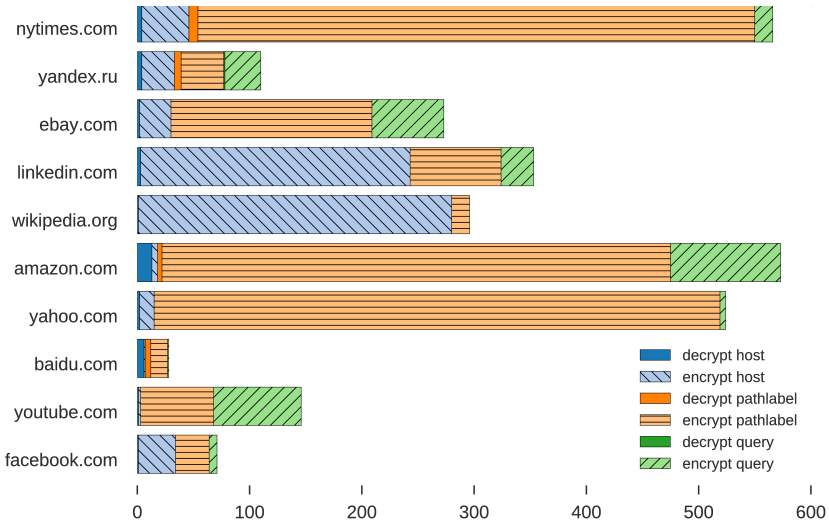


Figure 5.12: Encryption/Decryption operations for different web pages (Same Origin Policy content only)

By reviewing Figure 5.7, it can be seen that most websites only issue between 1 and 6 requests under the SOPs except *amazon.com* that performs 13 requests. Similarly the number of HTML URLs that fall under the SOP is also smaller (Figure 5.8).

For result analysis the strict definition of the SOPs is followed, meaning that different domains are not encoded. However it should be noted that this distinction is not always a clear indicator of origin. For example *nytimes.com* loads static content from the *nyt.com* domain, and *facebook.com* from *fbstatic-a.akamaihd.net* which are not rewritten since they fall outside the SOP. However this information is known to the service provider itself and could be articulated correctly into the policy for determining which URLs fall under the SBN.

Like before, Figure 5.12 shows the amount of decryption and encryption operations (per URL component) that were required to serve each of the analysed web pages. The most noticeable difference from the previous case (Figure 5.9) is a decrease in the number of decryption operations for the *Host* and *Path* URL components. Decryption of URLs happens when handling an HTTP request, and URLs in HTTP headers need to be converted. Since the number of HTTP requests that falls under the SOP is lower, the amount of decryption operations decreases proportionally.

Further optimisations of this nature would require internal knowledge about the content being served. For example the service provider may consider some URLs disclose no significant privacy information and keep them out of the SBN.

5.3.4.2 Encryption/Decryption Caching

This implementation already provides caching of encryption/decryption operations within the same HTTP request. However as pointed out earlier in Section 5.3.2.4

there is room for optimisation, if the server can cache encryption/decryption operations across multiple HTTP requests, but within the same session.

Continuing from the previous scenario, now extended with a cache for encryption/decryption operations, results in the operation counts seen in Figure 5.13. Unsurprisingly, there is a decrease in the number of encryption/decryption operations for all sites. These results assume a cache with no eviction, so these results offer a lower bound for this optimisation. With other caching policies, interference between different users using the same server would come into play.

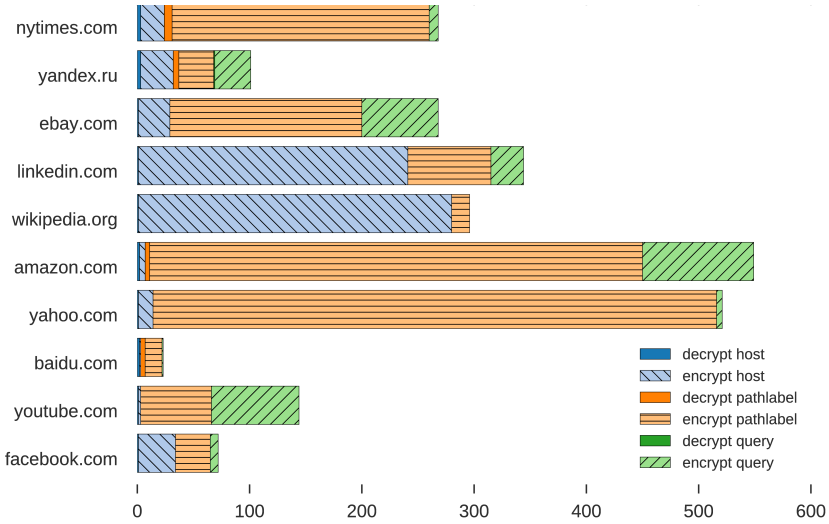


Figure 5.13: Encryption/Decryption operations for different web pages (SOP content only; With caching)

Some cases stand out where caching does not appear to be very effective. For example *linkedin.com*, generates HTML links using 238 different subdomains (af.linkedin.com, bb.linkedin.com, etc) causing a high number of encryption operations that cannot be cached. Similarly, Wikipedia uses 289 different subdomains. These two cases are particular in that hostname encryption represents the majority of operations Similarly *amazon.com* and *yahoo.com* also get only a slight improvement from caching, due to the high number of unique links in those pages. Naturally, the benefits of caching operations are less significant in these pages where most of the URLs are unique.

5.3.4.3 Collapsed Path encoding

From the previous results in Figure 5.13 it can be seen the majority of encryption/decryption effort is spent with URL path labels. Only in two particular sites (*linkedin.com* and *wikipedia.com*) this is not true, due to a high number of hostname TLDs in use. *Requirement 7* (defined in Section 5.3.1) requires path labels to be encoded separately. If this requirement is released, and the full path of the URL is

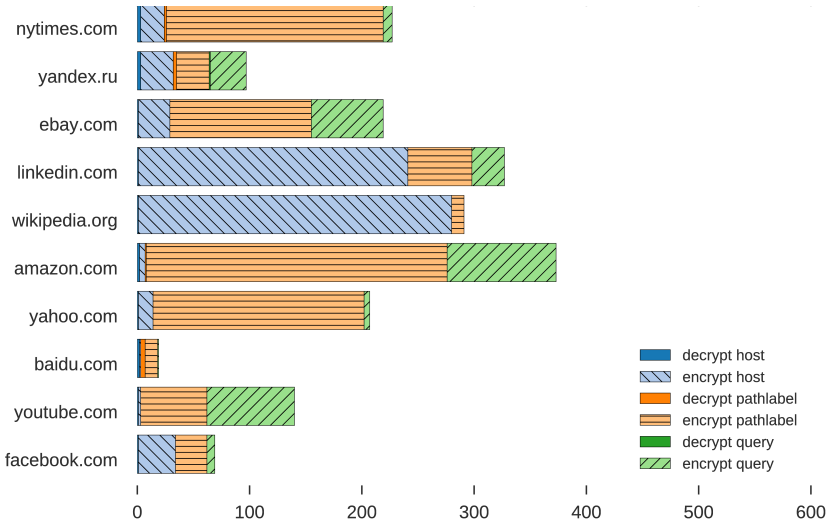


Figure 5.14: Encryption/Decryption operations for different web pages (SOP; caching; collapsed path encoding)

encoded in a single encryption operation (just like the *Host* and *Query*), the number of *Path* related operations would be a fraction of the previous case.

Like before, Figure 5.14 shows the amount of decryption and encryption operations (per URL component) that were required to serve each of the analysed web pages under this strategy. While the *Path* encryption operation count still looks high for *amazon.com* and *yahoo.com*, the number of HTML URLs in these two pages is 290 and 307 respectively. This means the results seen in Figure 5.14 are already under one path encryption operation per URL, due to the caching introduced in the previous step.

Recalculating the earlier delay estimates for the *amazon.com* case (370 encryption and 3 decryption operations) would require a delay of *11 ms* (Salsa20), *59 ms* (AES) and *1410 ms* (ECC), which is a significant improvement. The ECC values are still elevated, but are less than half of the previous values.

5.3.5 Deployment Considerations

Before closing off this topic, it is important to discuss the main barriers to the generalised adoption of these techniques, in particular those encountered while deploying the described implementation, and what approaches can be taken to address them.

The motivation for this approach is to provide service providers with mechanisms that enable them to improve privacy for URL namespaces under their control. This is a form of service enforced namespace privacy that mitigates attacks that target URL information, either in transit, via cache probing attacks, or through forensic inspection of URL caches in user devices.

However this technique has deployment requirements that need to be discussed as well as implications that must be considered before wide deployment can be sought.

5.3.5.1 Relation to other privacy tools

Mechanisms for confidentiality and anonymity are orthogonal to this approach. For example HTTPS would still be required for confidentiality as messages flow through the network. Likewise for source or destination anonymity ToR [203] can be used, replacing the use of regular DNS hostnames with an *.onion* address used to reach a hidden service, in which case the hostname in the URL is not altered.

For transient *.onion* addresses the service provider would need to setup multiple *.onion* addresses, much like is done here for DNS. The main disadvantage of ToR hidden services in the context of this scenario is that it requires server side setup of the ToR connection, as well as client side configuration (i.e. installing Tor), but the added anonymity may be necessary. Alternatively the user can access these services using proxy services, such as ToR2Web[11], at the expense of anonymity, and assuming such services are trustworthy. This is not surprising since the primary goal of ToR is network anonymity, while other client side tools (e.g. ToR browser) address privacy issues.

The Veil framework [205] implements multiple techniques for URL and page content obfuscation, that range from hiding URL metadata to remote browser execution [206]. It does not use URL encoding schemes like those described previously, instead it injects a javascript library in web pages to decode and fetch encrypted URLs. While it shares much of the same motivation, server side enforced privacy, the technical approach requires complete recompilation of all HTML/CSS/Javascript content application deployment and shifts some of the performance cost onto the client side. Primarily it is concerned with a different class of attacks, and the proposed techniques could be combined with the ones described here.

On the client side there are multiple tools for browser privacy protection, for example most browsers implement a private browsing mode, under which no browsing records are stored. However these are sometimes faulty, can be undermined by third party components [207, 208], or fail against forensic inspection [202]. The same can be said of privacy savy habits, such as regularly cleaning the cache and history, or using specialised extensions, but these are not practices the service provider can control or enforce on users. Finally, disk encryption tools can protect data privacy in stolen devices, but passwords can still be obtained through coercion, or poor judgement can lead to the use of a weak password.

5.3.5.2 URLs lose global meaning

Since URLs in the Session Bound Namespace (SBN) are transient and based on service provider policy, they loose their global meaning since they are only valid under a

[11] https://www.tor2web.org

certain scope. It is not unusual for service providers to generate transient links (e.g. temporary download links) but the generalised use of this approach as proposed here is not common.

The immediate implications of this change are a result of the browser/user being unaware of the transient nature of these URLs, as such:

1. Indexing of content (e.g. search engines) will not work
2. Bookmarking and sharing of URLs will not work

These aspects are significant usability issues that hinder user experience, but can be addressed through additional work, and seem desirable outcomes from privacy concerns.

Indexing of content by search engines is also covered in [173], which notes that search engines use well known user agents, and privacy mechanisms can be disabled for these indexers, if this is desirable. Bookmarking and sharing stop working because the URLs may have a limited life time, or worse, if they are pinned to the user session, a shared URL will always result in an error.

To approach this issue, further extensions to the SBN concept can be proposed. An example is sharing the encryption key with the client (e.g. as the session starts send the key as part of a cookie) allowing the browser to decode SBN URLs; However this potentially sacrifices the privacy of the namespace to browser implementation issues. For practical purposes this could be implemented as a browser extension. In simpler cases, like bookmarking or content sharing widgets, this can be implemented using javascript served as part of the web page. In effect this is the approach adopted by [205] for all cases.

While these usability issues may seem extreme, in a privacy dominated scenario they may be minor considerations. For example, in its most extreme case, [205] turns the browser into a remote client for rendering, forgoing any usability aspects.

5.3.5.3 Impact in DNS caches

Given that the SBN establishes transient DNS names for each session, the number of entries in DNS cache increases linearly with the number of sessions used by nearby users. In addition the size of each entry is usually large due to the base32 encoding.

In this particular instantiation, SBN URLs match user session (but other policies can be applied) and all DNS requests arrive from the same user terminal and there is little benefit to intermediate DNS caches outside the terminal. Assuming DNS caching resolvers apply a "least recently used" eviction policy, these entries are removed as soon as the cache becomes saturated. A non optimal case occurs for alternative eviction policies, because the cache may keep these transient entries when they are not needed and evict other entries instead.

5.3.5.4 Impact in HTTP caching

Given their nature as resource identifiers, URLs are used as keys for caching content, at various locations:

1. The client terminal, in order to reduce loading times.
2. The Service Provider, to reduce load on its infrastructure.
3. Transparently, at the network provider infrastructure, to reduce network load.
4. In leased infrastructure, close to the access network (e.g. CDN), where the service provider can place content to minimise retrieval times.

At the client terminal, the impact of this scheme will be inversely proportional to session duration. Short sessions can result in a cache filled with URLs that will not be reused. In fact, privacy conscious web sites could request a cache cleanup when logging out, minimising this issue, but so far no such mechanism is widely available in browser implementations.

Within service provider premises, the ability to revert the namespace mapping for incoming requests means content can be cached based on the original URL. If the mapping function is considered to be expensive, its results can also be cached to minimise retrieval delays as was discussed previously.

Transparent caches are those most affected by this scheme since, as a consequence, multiple copies of the same content can exist in the same cache. In the specific case of HTTP this is a known issue when serving user specific content that cannot be cached as a whole. To minimise this issue, the Service Provider may choose to serve some recurrent content outside of the SBN (sacrificing privacy for bandwidth).

Finally for service provider caches leased at the access network, the mapping can be reversed to retrieve from the cache. But this would require distributing the necessary keys through this infrastructure.

5.3.5.5 TLS deployment complexity

When transient hostnames are used, the necessary adjustments must be made to TLS deployment. Registration of TLS certificates in bulk, or even through wildcard hostname certificates, can be too expensive for some services. While free Certificate Authorities (CAs), such as letsencrypt[12], alleviate these costs they might not be desirable for the more privacy conscious services.

An additional usability concern is that the change into a strange hostname as the session start clashes with established security practices - users don't expect the hostname in webpages to change drastically as they log in, and look for the correct hostname for fear of phishing or TLS attacks.

[12] A CA which provides free wildcard certificates based DNS validation for three month periods.

It is also debatable whether extending this level of protection to the hostname is worth the cost and performance impact. Since the user always starts by using the real service hostname (step *1* in Figure 5.2), it can be argued a capable attacker will always be able to extract the hostname (using TLS Server Name Indication or DNS MITM attacks), unless other privacy tools are in place, such as ToR or DNSCrypt.

Economic alternatives to these issues are also worth exploring. If one service can't host multiple domains for SBN, maybe multiple privacy conscious services can agree on sharing a pool of subdomains for this purpose. One can easily imagine a *.sbn-sessions.org* service, that provides pools of hostnames for privacy conscious services to use.

5.3.5.6 Privacy caveats

While primarily motivated by privacy, there are a number of related caveats that are worth considering when using the proposed privacy strategy. First and foremost this type of approach is implemented by the service provider, and it is but an implementation of service provider policies. As a transitive side effect it may improve user privacy, but it would be naive for users to assume such policies to be anything but self serving (even if by legal requirements).

Some of the client side privacy protection mechanisms that were discussed earlier are meant for scenarios where the service provider itself, or some third-party is colluding to compromise user privacy. As such, SBN cannot replace these tools, as it can only assist the service provider in augmenting its own privacy.

Under the proposed scheme, URLs retain their original structure: the number of Path labels, presence of Hostname and Query are not changed, unless path encoding optimisations are considered. So far no privacy study addresses the possibility of uniquely identifying a service provider based on the structure of the URLs it generates i.e. if one inspects a browser cache solely based on URL structure rather than data and the URL relation graph, is there a significant probability of identifying a specific website or user? This is left as an open question, however the proposed scheme could be extended to randomise URL structure as a way to mitigate that possibility e.g. through the insertion of extraneous path labels, or simply by collapsing them into a single label.

The primary assumption is that transient URLs function as privacy preserving mechanism, and as such a valid concern is that the uniqueness of the URL components can be used to track the client. While the placement of unique identifiers inside URLs for tracking purposes is not new, unique session hostnames are not common, and will be "leaked" as the client terminal issues DNS requests. An attacker with the ability to intercept DNS traffic and a-priori knowledge about the SBN namespace, could exploit

this to track user movement, albeit with reduced metadata. The service provider can minimise this issue by forcing periodic rotation of the namespace/session, much like other network pseudonymity approaches (Section 5.4), otherwise the issue becomes more significant for long lived sessions.

5.3.6 Privacy policies and performance

The pros and cons of systematically applying URL transformation privacy have been considered, ranging from performance issues to deployment consideration (transferability, memorability, performance, setup costs).

The choice of which parts to deploy is ultimately a compromise between policy and cost. It is worth summarising here some of the main policies that were discussed in previous sections, and whether or not these should be adopted based on the previous results, since it is always possible to adopt a subset of these mechanisms:

Path: Protecting the URL Path alone will not prevent an attacker from determining which websites (hostnames) a user visited, but it does hide specific resources represented in the Path. In some cases this may be sufficient, if the Service Provider sees no need to conceal the remainder of the URL.

Hostname: From the points already discussed in this section, assigning new hostnames for different sessions may involve additional setup costs, and the privacy benefits are arguable without further considerations, e.g. the first contact will always disclose the hostname, and the leakage in DNS queries may be detrimental.

Query: Concealment of Query information through this scheme is highly dependent on the application, some service providers only use queries as part of client (browser) crafted URLs. But previous work suggests [174] that in general this component is worth concealing.

Mixed use: Using a mixture of bound and unbound URLs in the same website is also possible, provided the limitations of the Same Origin Policy are not an issue. It is already common behaviour for websites to offload popular website components (e.g. Javascript, images, videos) to external CDNs. Whether or not this is privacy leaking depends on the website.

The previous choices can be coupled with the implementation schemes and optimisations discussed in Section 5.3.4. In particular the following three are the most relevant:

1. The encryption scheme used to encode URLs
2. The number of encryption/decryption operations per URL
3. The effectiveness of caching encoding operations

The microbenchmark in Table 5.4 offers some indication on the delay introduced by different encryption schemes. In the presented example (Section 5.3.4.3), a delay of

10-30 ms (Salsa20) might be acceptable for websites that are not delay sensitive, but the *1-4 s* delay seen when using ECC is unacceptable for most cases.

An additional observation can be made that, for the described mapping function, the number of encryption operations is higher than the number of decryption operations (because the user might not open all encoded URLs). The choice of other encryption schemes, which are not covered by this work, could take advantage of this.

5.4 Pseudonymity mechanisms at the network layer

Fundamentally network pseudonymity consists in the use of distinct identifiers in different contexts. The notion of context is open to interpretation. A user might not want his actions to be correlated with what he does in his private time, which may span across time or through different locations. But the network has no notion of privacy, it routes packets regardless of user intent and a quick glance at the MAC or IP addresses in a packet allows one to infer if two packets are associated.

One of the goals of this work was to enable user centric privacy models and network pseudonymity was one way to achieve this. The work published in [34] is an implementation of network layer pseudonymity that provides mechanisms for a terminal to instantiate virtual wireless interfaces with distinct MAC and IP addresses. This enacts a type of pseudonymity, where an observer cannot correlate two streams of traffic as belonging to the same device based the source MAC and IP addresses. These pseudonyms were grouped into manageable contexts, or identities under which the user performed multiple activities (Section 5.4).

The core tenet of network pseudonymity is to hold distinct names for the various network related namespaces. This can be achieved through any kind of technique that establishes new bindings in these namespaces. First of all let us consider how this can be implemented, and then look at solutions to integrate this with higher layers and aligning high level concepts with a concrete implementation.

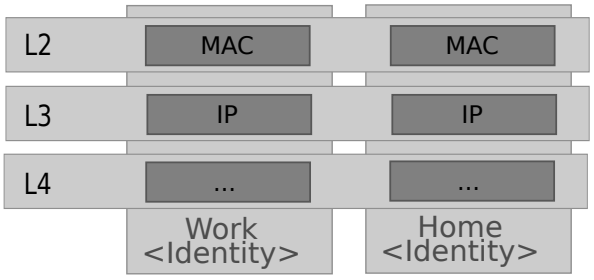


Figure 5.15: Cross layer identity contexts, establish groups of network pseudonyms.

5.4.1 Implementation

One form of abstraction that groups network identifiers are network interfaces, as seen in operating system resources. In essence, the solution implemented here is a Virtual Device Manager (VDM), a system service whose purpose is to manage virtual devices created over a physical network interface. This particular implementation targets the Linux operating system, and provides both control function to configure these interfaces as well as data functions to handle traffic forwarding through them. On the control path this service provides functions to instantiate new virtual interfaces with a different MAC and IP addresses over a physical network interface. On the data path, it handles packet switching from the underlying physical interface to the virtual interfaces and vice-versa.

For the control path it provides functions to create and delete new virtual network devices. These virtual devices are TUN/TAP interfaces, as provided by the Linux operating system. Each new interface is assigned a randomly generated MAC address, and must go trough normal network procedures to acquire an IP address (e.g. DHCP). Furthermore, new application processes can be bound to a specific virtual device. This is achieved through injection of shared libraries that intercept application calls to the socket API, and bind sockets on creation to the intended virtual interface.

Concerning the data path, the VDM acts similarly to a packet switch: it forwards packets between the virtual interfaces and the physical interface supporting them. Because the implementation is fully aware of MAC addresses for all the virtual devices, flooding is not required except when handling broadcast packets. This approach is similar to container based virtualization solutions, that have since become popular.

Physical support for this system includes both generic Ethernet devices and Atheros Wifi interfaces supported by the Madwifi driver. For wireless links, this was implemented using a modified wireless driver, through packet injection, which allows for multiple associations through the same card, provided they use the same channel. In effect this is the primary practical limitation to the adoption of these techniques in Wifi, because concurrent associations require driver or hardware support, which is not possible in all cards. The primary contributions of this thesis for this work targeted binding of interfaces to applications in the control path, and on packet switching on the data path. For this reason the experimental results discussed next focus on performance over Ethernet devices.

5.4.2 Alignment with upper layers

While [34] establishes the technical means for network pseudonymity. It still lacked the means to align this control to user notions of identity. [32] envisioned two ways to

do this, first for ad-hoc applications placed by the user under an identity label, and second in conjunction with external Identity Management (IdM) solutions.

For convenience the user can define local labels for user identities, which group applications during execution. An application under a specific identity context can only use the network pseudonyms associated to that particular context i.e. in practice it can only use a network interface created for that particular identity. Applications are started through a special interface that provides an identity selector, and binds applications to that specific interface. This context binding extends all new processes started under the chosen identity.

In addition, [35] extends this work to integrate it with external Identity Providers based on SAML. This means new identity contexts, and consequently network pseudonyms, are instantiated for different SAML identities. This match fits well, because identity management models already support the notion of pseudonyms [126, 96]. SAML defines different pseudonyms in its namespace, for each service provider the user contacts, and in turn these SAML pseudonyms are associated with a group of network pseudonyms

In effect, upper layer identities drives the assignment of pseudonym and creation of new virtual interfaces. This establishes a cross layer identity context that, in theory maintains distinct names in each context. However enforcement of this rule may require changes to existing applications, so they they can be instantiated under different identities. For example browsers executed under two different identities, used fully separate profiles with no common caches or extensions between them.

5.4.3 Integration with DNS privacy

For integration with DNS, [44] introduces a signalling mechanism for DNS resolvers to insert resolution hints into DNS queries. In general these resolution hints allow custom DNS resolution between components that support this extension, whose results are customised based on the provided hints. Specifically they are used in [209], for two purposes: to implement DNS authentication through a side channel with push based authentication protocols, and to apply custom name resolution on top of the DNS protocol.

The implemented approach follows techniques similar to those of [210] or [211], in that a nameserver specific hint is placed in DNS requests and responses as an *OPT* record (Figure 5.16). Internally the hint is represented as a regular TXT record [179]. This hint can be inserted by individual applications that choose an application specific hint, or for the shared resolver at the user terminal.

In [209], this mechanism was reused to implement access control for DNS queries, where the resolution response depended on the identity of the user making the request.

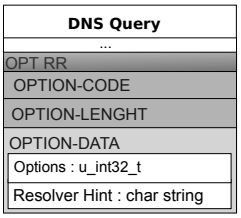


Figure 5.16: Resolution hint option in DNS messages (grey fields are EDNS0 headers)

For this particular case the hint placed in the DNS query was a user identifier, or rather a pseudonym from an IdM provider.

The implementation in [44] includes three components: a resolver library to issue DNS queries based on the Unbound [102] library; a modified authoritative DNS nameserver based on [212], and an authentication agent to handle authenticated queries.

If the nameserver does not support this extension, the query fails, yielding a DNS error reply, as defined in RFC2671 [213]. A new query can then be made without a hint, to get a regular answer, this works as a feature detection mechanisms where the resolver library caches this information. For nameservers that support the extension, the additional header is also used to carry error conditions that relate to the hint. By itself this extension does not address confidentiality. This is left for other solutions such as [46, 108], that already address this problem within a limited scope.

Like in [210] this approach must be used carefully, because an intermediate resolver would cache the responses to these queries which might lead to undesirable side effects. Instead this is meant to be used between a client with a specialised resolver library and a supported nameserver, or between two resolvers (during iterative resolution) that support this extension in order to convey specialised resolution context. This is its primary downside, since it forgoes caching in intermediate DNS caches (if they do not support this extension), otherwise the query responses might be cached and applied to other queries for the same name.

For experimental testing in [44] a self contained implementation that uses XMPP for authentication signaling [214] is used instead (Figure 5.17). In this case the authentication process may introduce enough delay that cases the initial resolver to consider the query message was lost and issue a new query. Nameserver query timeout usually implements a backoff algorithm that can go up to 45s [215]. This is more than sufficient. But duplicate DNS queries may occur between the resolver and the nameserver, in which case the nameserver identifies them as being identical.

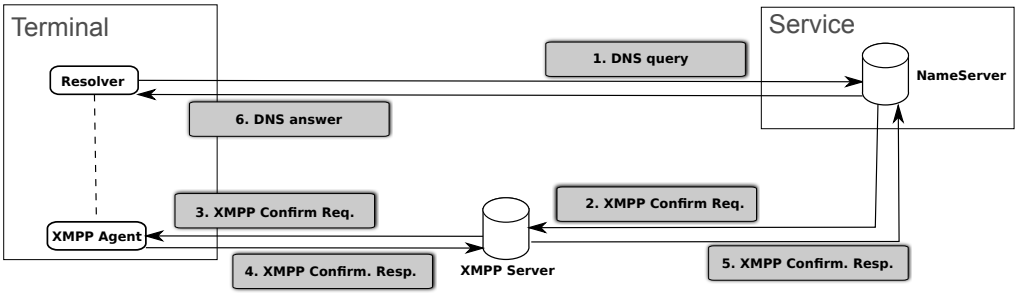


Figure 5.17: Authenticating DNS queries using a XMPP authentication extensions

5.4.4 Performance impact

To analyse the impact of this implementation three metrics were selected based on impact on network operations and setup times. The first was communication delay, which is measured using UDP traffic; the second was available bandwidth measured using TCP flows; and the third was the necessary time to bootstrap a new interface including normal network registration (i.e. DHCP configuration and system setup). All presented data points are averages of 15 runs, with a 99% confidence interval.

For comparison, reference values measured without this implementation are also provided. When testing with multiple UDP/TCP flows, the reference case uses multiple flows over the same physical interface while the VDM uses one flow per virtual interface.

Figure 5.18, shows an average delay penalty of 36ms. The additional delay ranges from 20% to 50% of the total delay, and as the number of interfaces/flows increases so does the delay. This is also true for the reference case.

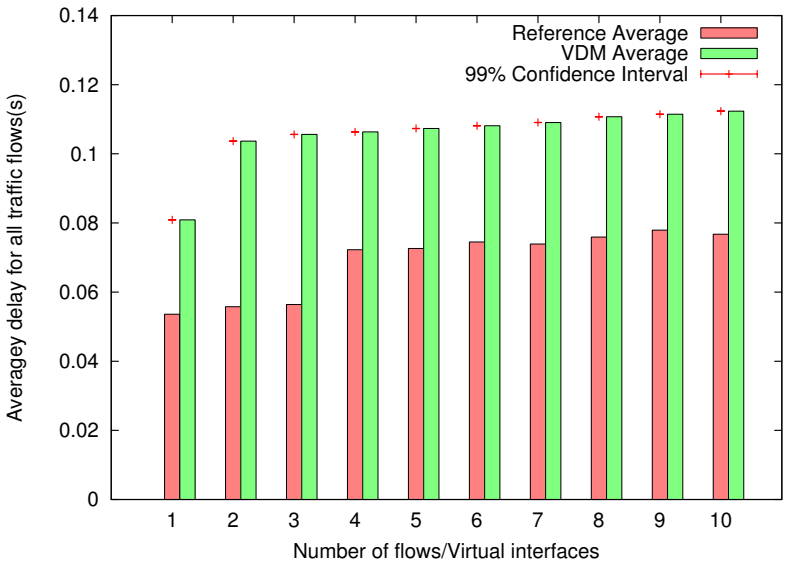


Figure 5.18: Communication delay for UDP

There is room for improvements on this metric, through changes to the current implementation. Since these interfaces are implemented as Linux TUN/TAP interfaces, and internally with packet switching using user-space buffers, a number of memory copies are required for moving packets through. A production grade implementation would require moving to a kernel only implementation, that could improve this by avoiding copying packet buffers and only sharing memory pointers.

Available TCP bandwidth (per flow/interface) is presented in Figure 5.19. In the reference case, multiple flows (through a single interface) are also used. As expected it can be observed that bandwidth is affected. In particular the average bandwidth decreases as the number of network interfaces increases. For clarity, the difference between the reference measurements and the available bandwidth when using multiple virtual interfaces is also depicted.

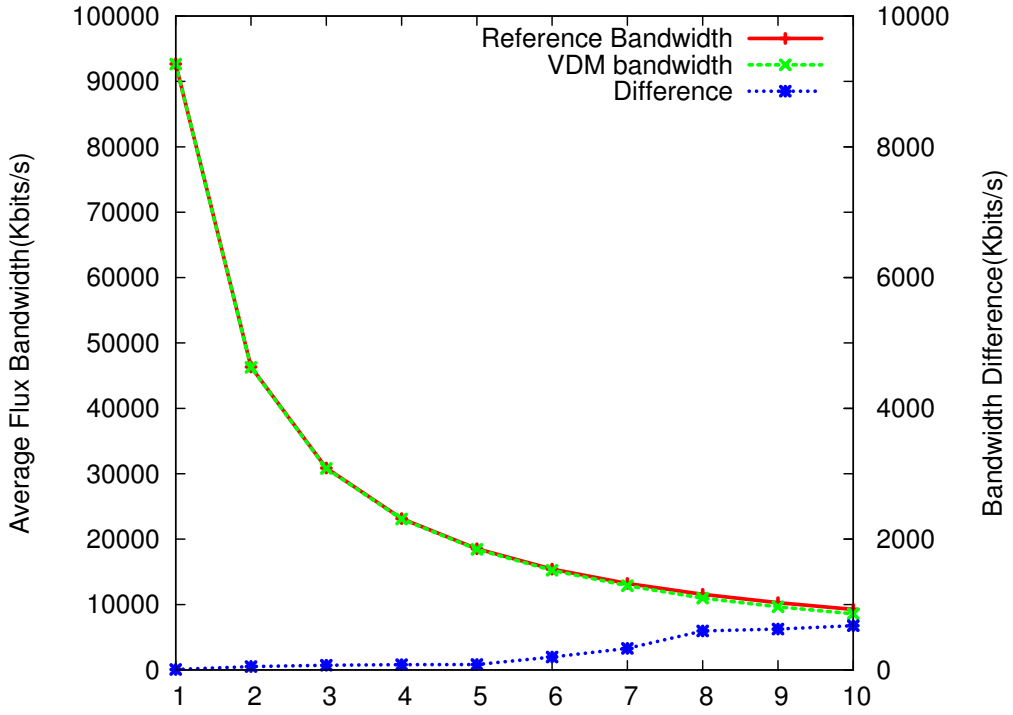


Figure 5.19: Average TCP bandwidth per flow/interface.

Bootstrapping of new interfaces, shown in Figure 5.20, takes an average time of 3 seconds. This time includes all the necessary network association procedures, since the interface is created until connectivity is available. As can be seen, this time does not depend on the number of previously existing interfaces.

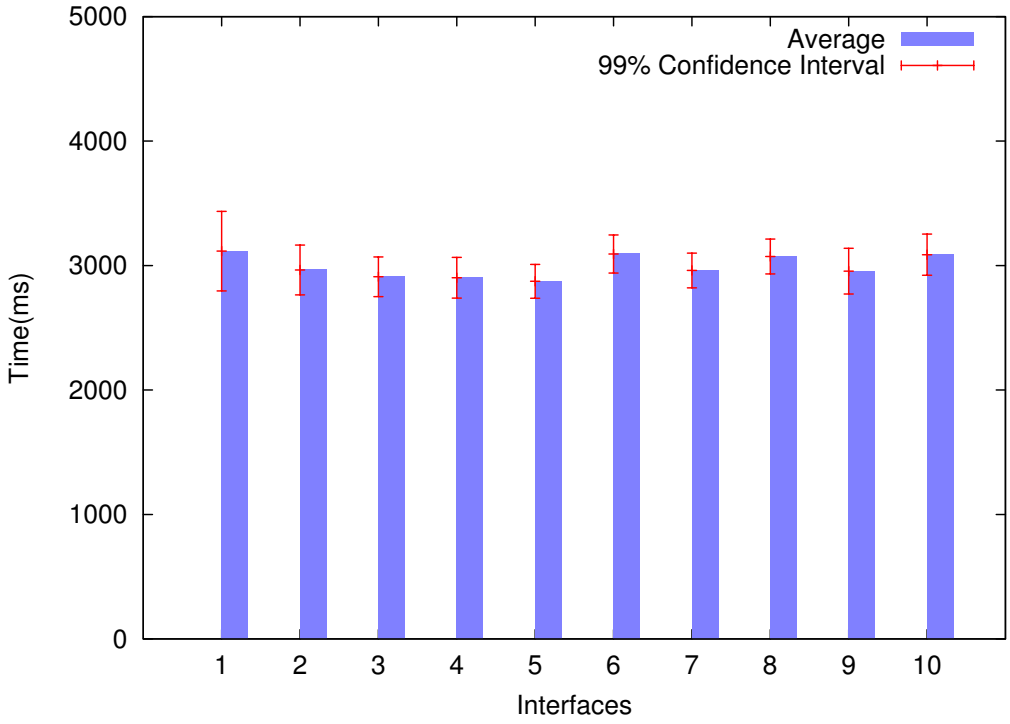


Figure 5.20: Bootstrap delay of several virtual interfaces.

5.4.5 Limitations

Given that a single device may operate multiple pseudonyms simultaneously, this reduces the number of available names in the network. Even assuming the MAC and IPv6 namespaces support up to 2^{48} and 2^{64} devices respectively, and will not be exhausted within the local network scope, it is still possible for naming collisions to occur for randomly generated values. In practice, to keep the collision probability below 0.1%, the maximum number of simultaneous virtual devices is restricted to 5. If collisions still occur, mechanisms such as IPv6 Duplicate Address Detection [18] need to be used to detect them, and the interface will need to be reset. Currently most operating systems already provide similar mechanisms to detect address duplication.

As one shifts away from the scope of the local network, pseudonymity approaches present different requirements. Since network addresses are assigned localy, the client has some leeway with regards to the assignment of multiple addresses (pseudonyms). If needed, the same terminal can host multiple network cards, but as previously seen this is not strictly necessary. Once multiple real, or virtual network interfaces are in place, then assignment of multiple IP addresses becomes possible. This is the basis for pseudonymity solutions, and it works for the source address because the client can influence the scope where it is assigned, provided the local network allows it. Likewise at the upper layers, it works for IdM solutions such as SAML because the user can

instantiate per service pseudonyms exactly for this purpose.

However the same cannot be said for names completely outside the control of the user. It might be impossible for a user to bind a name to use as a pseudonym in such namespaces. Or the binding of such name could have unintended consequences. For example binding a random DNS name to use as a pseudonym to a service could be locally meaningful, but could never be used in HTTP, because an HTTP service will not respond to a host name it does not own [134].

As such, to conceal information on names outside local control, the user must still resort to solutions such as ToR or VPNs that provide confidentiality through a third-party that provides an endpoint. Moreover all these solutions are heavily user centric, and require the use of custom components by the user and/or support from an intermediate party. Going forward into the next section, the assumption is that this is not always a possibility and that privacy at the upper layers may be breached either through application side channels that reveal too much information, or from user action that unintentionally bridges the gap.

5.5 Conclusions

In this chapter, mechanisms for improving naming privacy were introduced in multiple scopes, making an transversal cut through the network stack.

At the application layer, it can be seen that privacy disclosure arrives in multiple forms. Sometimes the underlying protocols are designed in such a way that immediately discloses identifiers from some namespace, in others protocol design is undone by implementation or poor user choices. Section 5.2 takes a glance at a particular instance of this problem, and finds some cases of privacy disclosure that result from the cross-over between XMPP and HTTP, in particular that some techniques used to attack HTTP produce interesting results for XMPP.

Borrowing from the state of the art it can be seen that a significant part of namespace privacy work is implemented through client side extensions. This works well up to a point, but the assignment of names in this context is not related to the user but rather to service provider policy. As such it is worth moving to the study of mechanisms that increase privacy in the assignment of URLs.

With regards to general concealment of information in the URL namespace (Section 5.3), privacy comes at a cost, that needs to be carefully balanced with the intended privacy requirements. Not all measures are equally effective. Generating alternative DNS hostnames implies setting up DNS infrastructure and TLS certificates beforehand, which may be be costly and not fully effective. Per label path encryption can quickly

become costly and the stringent requirements which were introduced for compatibility may have to be relaxed to address this issue.

Ultimately this approach provides a range of options to find an effective compromise for service provider enforced privacy. Some other approaches are left unexplored, but may be worthwhile as pursuing as applied to specific cases, or as a refinement to the discussed approaches. Alternative encryption schemes, or precomputation of parts of the namespace prior to use instead of relying on caching on-demand.

Finally pseudonymity approaches for IP and MAC addresses were introduced in Section 5.4. These are effective as a client side mechanism for preventing correlation of information based the terminal addresses. For wide applicability they are somewhat dependent on feature support in wireless devices, or conversely the number of pseudonyms that can be held in parallel is limited by device support. These mechanisms can be tied to different control mechanisms, based on user control or from other notions of pseudonymity from IdM protocols.

It is worth noting here that the high level concepts studied here do not exhaust themselves within this context, they are equally applicable in other protocols. Some of the techniques described in this chapter could be equally applied to some of the ICN protocols seen earlier, and similar contributions can be seen in that context NDN[161].

Chapter 6

Conclusions

We're in the endgame now

Dr. Stephen Sanders

As the final chapter of this thesis, it is time to take a step back, provide an overlook on how what was done ties together, review the initial objectives and point out future directions.

6.1 Results & Achievements

This thesis introduces multiple mechanisms that alter name assignment policies in multiple namespaces as one way to achieve improvements on security and privacy. The assignment of names can be leveraged for features which are orthogonal to unique identification or even to name resolution. At the genesis of this PhD thesis are two specific topics that revolve around name assignment. The first is the embedding of information in names, and the second is privacy disclosure as a consequence of naming policies.

For this work, the implementation targets assumed commonly used protocols and services. This keeps us grounded in current practices while keeping an eye out for similar trends in alternative spaces, but it takes us into a mixture of unspecified behaviour and current practices. In other words Saltzer[54] was right, in that the meaning of a name, even the same name, is highly dependent on context and this characteristic is commonly exploited for all kinds of purposes.

First, this thesis sets out to introduce secure namespaces into discovery protocols. This can be seen as a natural progression for HIP [10], as the benefits that follow are not dissimilar. But its applications fall onto a different type of scenarios, decentralised networks and point to point communications. From there, other types of security information, signatures and even certificates can be part of names in some of

these protocols. Unstructured networks can benefit from such a global namespace, for implementing protocols and services that are independent of the underlying network.

Since network namespaces tend to grow in usage scope, and be composed onto other namespaces privacy becomes increasingly important. For this reason, techniques for increasing privacy in several namespaces are implemented and evaluated. As a side effect of this process, additional mechanisms for privacy exploitation were also proposed.

The sub-sections that follow derive conclusions on the main subjects of study in this thesis. The order is similar to that of the structure of the overall document.

6.1.1 Secure binding in discovery namespaces

Chapter 3 embeds security information in different discovery namespaces. This information is then used to bootstrap security mechanisms, based on public key cryptography. Through this approach one can introduce security features even if the underlying discovery protocols lacks support for then, embedding hashes and signatures that can be used for verification in the corresponding namespaces.

Different namespaces exhibit distinct properties and the type or amount of information that can embedded varies, as does the resulting security capabilities. But at the very least, some namespaces can carry public key based authentication for protocols such as Bluetooth, DLNA and DNS-SD. Presumably this can be extended for other discovery namespaces that employ URLs as names. This also extends to namespaces that hold similar properties. Other network architectures, (namely NDN) are targeted in this context, but similar network architectures could also be used. A limiting factor here is the need to announce/publish additional names, to carry this information or a minor impact on the cost of service advertisement. Since the fundamental protocols remain unchanged, these mechanisms cannot prevent a Denial of Service Attack (DoS) attack, but they shift security verification procedures to the discovery process and increase privacy protection for the initiating party.

Hash based names are in widespread use in multiple technologies. They provide a common approach for data integrity verification. But they are also used as identifiers for key verification. Because the identifier is derived from a public key, this results in globally unique (with low probability of collision) names. With this property in place, and with the derived security mechanisms, other features can be implemented on top of these discovery protocols.

6.1.2 A global discovery namespace

From a global discovery namespace, with security capabilities, some forms of session mobility can be achieved. In particular, this is a form of cross protocol mobility, where a

previously known endpoint can be reached through a different transport protocol, which enables policies that favour power consumption or throughput according to desired policies.

Furthermore the availability of a cross protocol discovery namespace provides other features that do not rely on the presence of security information in its names. A global service type namespace facilitates cross protocol scenarios. Likewise, a global device identification namespace facilitates network interoperability. In particular this is explored in the construction of IoT discovery gateways, since they do not have to hold state.

However it needs to be recognised that, like in previous cases in the literature, the creation of a global namespace entails privacy risks. In particular for the case of a globally unique device or key names, these are prone to mobility tracking by third parties, or from collusion of involved parties. Some of the discussed privacy techniques are after all a reaction to this type of concerns.

6.1.3 Privacy at the network layer

The notion of protecting privacy at the network layer, through allocation of multiple names (i.e. pseudonymity) is a straightforward reaction to avoid disclosure of information in those namespaces, in particular to prevent identification across different scopes. New names are allocated to limit privacy exposure.

Names in network namespaces are usually finite resources. Thus privacy through these method has a cost in network availability since scarcity may prevent attachment to the network. Ultimately these resources are owned by the network provider, which may not be sympathetic to user privacy concerns, leading the user to resort to other types of privacy solutions.

At the network layer this is usually considered to be a problem with hostile observers. But the shift proposed in ICN could change all this, leading to a privacy scenario that is directly affected by application policy.

6.1.4 Pseudonymity shift at the upper layers

When considering upper layer protocols, the primary distinction with regards to the network namespaces is that users hold no bearing in the assignment of names. As such with regards to this thesis the focus is on privacy mechanisms introduced by the service provider, in particular this is discussed in the conext of HTTP service providers.

This does not mean client side privacy solutions do not exist, or are not required. It only implies that privacy mechanisms implemented through specialised name assignment fall under the operational responsibility of the operators that assign those names.

The primary means to achieve this is through encryption of individual URL components, to generate transient reversible names. Since URL components have different types, and each its own scope, this process is split over multiple namespaces and protocols. DNS hostnames need to be allocated in the corresponding DNS nameservers, and TLS certificates need to be generated accordingly. Path segments are encrypted labels that are only meaningful to the service provider. But due to their structure, clients expect to manipulate them accordingly. As such strict requirements should be introduced to allow this.

It is arguable that the strictness of the requirements introduced in this application-oriented implementation can be relaxed through client side support, either at the expense of some privacy or through development tools that address individual shortcomings. [205] provides an approach at the opposite end of the spectrum, with high privacy requirements, but shifting most of the performance costs onto the client side or additional deployment tools. But we attempt to mimic the on the digital world, the privacy notions of the physical world, the creation of *private spaces* which are held as such by its owner.

Looking at common purpose encryption implementations, this is still the bottleneck for practical use of this approach. As such the degree of privacy needs to be carefully weighted against the performance penalty. Much like the current trends on TLS adoption by service providers, it might be worth considering that this type of privacy disclosure warrant a similar investment. But such a degree of granularity requires a clear definition on which names reveal too much, and which do not. This implies, some type of detailed data model is in place that accurately categorises names (or perhaps the objects they bind to) into privacy labels.

6.1.5 Revisiting Hypothesis and Objectives

At the start of this thesis, there was a question on how to assign names, and the consequences of doing it differently. To study this question one had to look at established practices, because most resolution systems are (to some degree) amenable to different policies on name assignment.

To some degree, it is possible to add security semantics in names from common purpose discovery protocols. The primary limitation being the amount of information that can be embedded in a name, as well as protocol specific implications on use and performance. An interesting consequence of doing this is that a namespace can be defined for cross protocol identification of services and devices, a powerful tool for interoperability. A consequence of this achievement is that, as any global namespace, it may be too privacy revealing.

For privacy purposes, the main technique covered in this thesis is that of

pseudonymity, the transient assignment of names. The concept is straightforward but the implications of its implementation vary with each namespace. Going up the stack, or rather moving from local scope to distributed services over a global scope implies a shift in the techniques used to achieve it.

The conclusion, at this point, is that the hypothesis was validated under the presented results. Names can be bound in ways that satisfy security and privacy goals. But not without implications, on deployment considerations and a myriad of factors that warrant discussion and further study going forward. Rather than a yes or no answer, what is provided here is a set of techniques to implement such approaches, and some tools to evaluate the cost of privacy by these techniques.

With regards to privacy requirements, it is important to remember that privacy is not a unanimous concept. Often actual privacy is a consequence of enforcement of multiple (sometimes contradicting) privacy policies. In particular, one type of privacy protection does not relax the need for other complementary mechanisms.

More than the individual techniques or the solutions implemented here, it is the pinning of this recursive relation that should be kept in front of one's mind: that as the network grows, global namespaces tend to emerge for the purposes of identification, but must be followed by privacy mechanisms to curve this trend.

6.2 Future Work

Based on some of the conclusions from this chapter, but also on open points from the previous chapters it is possible to draft some upcoming topics of interest, that either flow as a direct consequence of this thesis or reflect related areas that were studied throughout this work.

6.2.1 Namespace composition, routing and nesting

The overview presented in Chapter 3 shows how many systems define their namespace as a composition of names from other namespaces. URLs are perhaps the most common example, using a DNS hostname or IP and a file path.

XRI takes a different approach that generalises the nesting of names in other names. The primary difference in XRI is that it allows for the inclusions of names in the XRI namespace, i.e. a name may depend on other names in the same namespace.

XIA takes another approach and extends the notion of route as a name that is constructed as sequence of names. It does this by using a Directed Acyclic Graph (DAG) as a name. Instead of expressing a route, a name can be considered to represent a list of possible routes, or a list of vertices in a graph.

It seems likely this type of construct could be explored for purposes other than the ones it was originally intended for. In particular DAGs are a common way to express distributed protocols, and privacy aspects could be pinned here.

6.2.2 Adversarial Privacy

It was already seen previously that relations of privacy can be transitive. One user may reveal another's location, or a service may expose identifying information that reveals user identity.

An aspect pointed at in Section 5.3.5.6 is that privacy improvements for one party may end up decreasing privacy for another. This turns privacy into an adversarial problem, not because an attacker attempts to compromise it, but because privacy efforts by one party may hinder another.

While there are models for privacy exposure, it would be interesting to close in on the characteristics of these particular cases as targets for early elimination, in particular if the privacy goals of multiple parties are not contradictory. This would mean finding the overlap of privacy features that are not detrimental to other parties, and eventually this could be seen as a distributed protocol on privacy agreement.

6.2.3 Applications outside of the scope of this thesis

This thesis draws heavily from current trends in ICN namespaces, and in particular it studied them as used in alternative network architectures (e.g. NDN). Some of the techniques introduced in Section 5.3 could be easily implemented in those architectures.

Related work in this context is embodied by [27, 216] and shows that the similarities between URLs and NDN names are used to facilitate interoperability in this context, while privacy mechanisms are introduced to avoid active censorship at the network layer, because in NDN these names are fully visible by eavesdroppers. More generally, the techniques in Section 5.3 can be used for any URL in any protocol. However, the requirements used in this particular implementation with regards to key distribution might have to be revised. In other scenarios, the SBN may be defined based on keys shared by client and service (defining a private namespace), or multiple services (a form of transient reference passing similar to SAML).

Based on Section 5.2 the use of browser fingerprinting for breaching privacy in other protocols seems promising, even more so considering protocols with mixed use of the HTTP protocol.

Finally it is worth pointing out that to some extent the global namespace defined in Section 4.2 is composable with the privacy techniques described in sections section 5.4 and section 5.3.

6.3 Final Thoughts

The topics of security and privacy now gather significant attention, primarily because of the wide spreading impact they have in society. Not only are individuals more aware of the lack of privacy and its consequences, but companies now need to weight the value of privacy against liability and consequences of exposure. But any considerations about the cost/benefit of security and privacy are hard to support a-priori. Either it is considered early on as a design requirement, or often too late to be effective.

In this thesis, these two aspects are studied within the context of naming. First as a means to provide security features, and then as a privacy challenge to be tackled. In retrospective this process of defining namespaces for security and then redefining them as scope increases to preserve privacy seems to repeat (over time), and will ultimately repeat.

The naive approach is to ignore this, and assume there will always be enough resources to continue repeating this process. But, ideally, one should aspire to keep this notion in mind before drafting a new namespace.

Furthermore since namespaces are composed from multiple sources the properties of names seem to be in permanent flow, as conflicting requirements clash over time. Privacy requirements are not uniform across the various involved parties. Some will value their other features above privacy (e.g. global uniqueness), or their privacy at the expense of others.

Bibliography

[1] Andrew Tanenbaum and Maarten Van Steen. *Distributed systems*. Pearson Prentice Hall, 2007.

[2] Drummond Reed and Dave McAlpin. *Extensible Resource Identifier (XRI) Syntax V2.0*. Organization for the Advancement of Structured Information Standards (OASIS), Committee Specification. Nov. 2005. URL: http://www.oasis-open.org/committees/download.php/15376.

[3] Venugopalan Ramasubramanian and Emin Gün Sirer. "The design and implementation of a next generation name service for the internet". In: *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. Portland, Oregon, USA: ACM, 2004, pp. 331–342. DOI: 10.1145/1015467.1015504.

[4] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. "A data-oriented (and beyond) network architecture". In: *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. Kyoto, Japan: ACM, 2007, pp. 181–192.

[5] *The FP7 4WARD Project*. URL: http://www.4ward-project.eu/.

[6] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. "Networking Named Content". In: *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. CoNEXT '09. Rome, Italy: ACM, 2009, pp. 1–12. DOI: 10.1145/1658939.1658941.

[7] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy kc, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. "Named Data Networking". In: *SIGCOMM Comput. Commun. Rev.* 44.3 (July 2014), pp. 66–73. DOI: 10.1145/2656877.2656887.

[8] Emmanuel Baccelli, Christian Mehlis, Oliver Hahm, Thomas C. Schmidt, and Matthias Wählisch. "Information Centric Networking in the IoT: Experiments with NDN in the Wild". In: *Proceedings of the 1st ACM Conference on Information-Centric Networking*. ACM-ICN '14. Paris, France: ACM, 2014, pp. 77–86. DOI: 10.1145/2660129.2660144.

[9] Ravi Ravindran, Yanyong Zhang, Luigi Alfredo Grieco, Anders Lindgren, Dipankar Raychadhuri, Emmanuel Baccelli, Jeff Burke, Guoqiang Wang, Bengt Ahlgren, and Olov Schelen. *Design Considerations for Applying ICN to IoT*. Internet-Draft draft-irtf-icnrg-icniot-01. Work in Progress. Internet Engineering Task Force, Feb. 2018. 50 pp. URL: https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-icniot-01.

[10] R. Moskowitz, T. Heer, P. Jokela, and T. Henderson. *Host Identity Protocol Version 2 (HIPv2)*. RFC 7401. Apr. 2015. URL: http://www.rfc-editor.org/rfc/rfc7401.txt.

[11] Thomas R. Henderson, Christian Vogt, and Jari Arkko. *Host Mobility with the Host Identity Protocol*. RFC 8046. Feb. 2017. DOI: 10.17487/RFC8046. URL: https://rfc-editor.org/rfc/rfc8046.txt.

[12] Pekka Nikander and Julien Laganier. *Host Identity Protocol (HIP) Domain Name System (DNS) Extensions*. RFC 5205. Apr. 2008. URL: https://rfc-editor.org/rfc/rfc5205.txt.

[13] Jeff Ahrenholz. *Host Identity Protocol Distributed Hash Table Interface*. RFC 6537. Feb. 2012. URL: https://rfc-editor.org/rfc/rfc6537.txt.

[14] I. Baumgart and S. Mies. "S/Kademlia: A practicable approach towards secure key-based routing". In: *2007 International Conference on Parallel and Distributed Systems*. Dec. 2007, pp. 1–8. DOI: 10.1109/ICPADS.2007.4447808.

[15] Célestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. "Defeating MAC Address Randomization Through Timing Attacks". In: *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '16. Darmstadt, Germany: ACM, 2016, pp. 15–20. DOI: 10.1145/2939918.2939930.

[16] Simon Josefsson and Linus Nordberg. *Improving Privacy for the email Received Header*. Internet-Draft draft-josefsson-email-received-privacy-01. Work in Progress. Internet Engineering Task Force, Nov. 2015. 5 pp. URL: https://datatracker.ietf.org/doc/html/draft-josefsson-email-received-privacy-01.

[17] T. Berners-Lee, R. Fielding, and H. Frystyk. *Hypertext Transfer Protocol – HTTP/1.0*. RFC 1945. Internet Engineering Task Force, May 1996, p. 60. URL: http://www.rfc-editor.org/rfc/rfc1945.txt.

[18] S. Thomson, T. Narten, and T. Jinmei. *IPv6 Stateless Address Autoconfiguration*. RFC 4862. Sept. 2007. URL: http://www.rfc-editor.org/rfc/rfc4862.txt.

[19] T. Narten, R. Draves, and S. Krishnan. *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. RFC 4941. Sept. 2007.

[20] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. "IP geolocation databases: unreliable?" In: *SIGCOMM Comput. Commun. Rev.* 41.2 (Apr. 2011), pp. 53–56. DOI: 10.1145/1971162.1971171.

[21] Russ Housley, John Curran, Geoff Huston, and David R. Conrad. *The Internet Numbers Registry System*. RFC 7020. Aug. 2013. DOI: 10.17487/RFC7020. URL: https://rfc-editor.org/rfc/rfc7020.txt.

[22] Chuanxiong Guo, Yunxin Liu, Wenchao Shen, H.J. Wang, Qing Yu, and Yongguang Zhang. "Mining the Web and the Internet for Accurate IP Address Geolocations". In: *INFOCOM 2009, IEEE*. Apr. 2009, pp. 2841–2845. DOI: 10.1109/INFCOM.2009.5062243.

[23] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos. "A Survey of Information-Centric Networking Research". In: *IEEE Communications Surveys Tutorials* 16.2 (Feb. 2014), pp. 1024–1049. DOI: 10.1109/SURV.2013.070813.00063.

[24] Sebastian Kaebisch and Takuki Kamiya. *Web of Things (WoT) Thing Description*. Tech. rep. First Public Working Draft. W3C, Sept. 2017. URL: https://www.w3.org/TR/wot-thing-description/.

[25] *IPSO Alliance, "Enabling the Internet of Things"*. URL: http://www.ipsoalliance.org.

[26] Zach Shelby, Klaus Hartke, and Carsten Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. June 2014. DOI: 10.17487/RFC7252. URL: https://rfc-editor.org/rfc/rfc7252.txt.

[27] Reza Tourani, Satyajayant Misra, Joerg Kliewer, Scott Ortegel, and Travis Mick. "Catch Me If You Can: A Practical Framework to Evade Censorship in Information-Centric Networks". In: *Proceedings of the 2Nd ACM Conference on Information-Centric Networking*. ACM-ICN '15. San Francisco, California, USA: ACM, 2015, pp. 167–176. DOI: 10.1145/2810156.2810171.

[28] Abdelberi Chaabane, Emiliano De Cristofaro, Mohamed Ali Kaafar, and Ersin Uzun. "Privacy in Content-oriented Networking: Threats and Countermeasures". In: *SIGCOMM Comput. Commun. Rev.* 43.3 (July 2013), pp. 25–33. DOI: 10.1145/2500098.2500102.

[29] Rui Ferreira, Alfredo Matos, Susana Sargento, and Rui L. Aguiar. "Enabling Identity Aware Applications". In: *Proc Conf. sobre Redes de Computadores - CRC*. Oeiras, Portugal, Oct. 2009.

[30] Eric Rescorla and Tim Dierks. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. Aug. 2008. DOI: 10.17487/RFC5246. URL: https://rfc-editor.org/rfc/rfc5246.txt.

[31] Azevedo, R., ed. *FP7 SWIFT, Deliverable 207b, Final SWIFT architecture*. Apr. 2010.

[32] Rajasekaran, H., ed. *FP7 SWIFT, Deliverable 207, Final SWIFT architecture*. Apr. 2010.

[33] Matos, A., ed. *FP7 SWIFT, Deliverable 204, Identifiers and Name Resolution Namespaces, Discovery and Federation*. 2008.

[34] Alfredo Matos, Rui Ferreira, Susana Sargento, and Rui Aguiar. "Virtual Network Stacks: From Theory to Practice". In: *Wiley Security and Communication Networks* 5.7 (July 2012), pp. 738–751. DOI: 10.1002/sec.368.

[35] Rodolphe Marques, Rui Ferreira, and Alfredo Matos. "Cross Layer Privacy Support for Identity Management". In: *Future Network and Mobile Summit*. MS10. Florence, Italy, June 2010.

[36] Rui Ferreira, Alfredo Matos, Goncalo Morais, Rui L. Aguiar, Pedro Santos, and Ricardo Pereira Azevedo. "Multipass: Gestão de e-Tickets em Dispositivos Móveis". In: *Revista Saber & Fazer Telecomunicações* 9 (Dec. 2011), pp. 76–81.

[37] Rui Ferreira, Alfredo Matos, Susana Sargento, and Rui L. Aguiar. "Multipass: Autenticação Mútua em Cenários Heterogéneos". In: *Proc Conf. sobre Redes de Computadores - CRC*. Aveiro, Portugal, Nov. 2012.

[38] Rui Ferreira, Alfredo Matos, and Rui Aguiar. "Recognizing Entities Across Protocols with Unified UUID Discovery and Asymmetric Keys". In: *IEEE GLOBECOM*. 2013.

[39] José Quevedo, Rui Ferreira, Carlos Guimarães, Rui L. Aguiar, and Daniel Corujo. "Internet of Things discovery in interoperable Information Centric and IP networks". In: *Internet Technology Letters* 1.1 (2018). e1 ITL-17-0001.R1, e1–n/a. DOI: 10.1002/itl2.1.

[40] José Quevedo, Carlos Guimarães, Rui Ferreira, Daniel Corujo, and Rui L. Aguiar. "ICN as Network Infrastructure for Multi-Sensory Devices: Local Domain Service Discovery for ICN-based IoT Environments". In: *Wireless Personal Communications* 95.1 (July 2017), pp. 7–26. DOI: 10.1007/s11277-017-4425-7.

[41] Daniel Corujo, Carlos Guimarães, José Quevedo, Rui Ferreira, and Rui L. Aguiar. "Information Centric Exchange Mechanisms for IoT Interoperable Deployment". In: *User-Centric and Information-Centric Networking and Services: Access Networks and Emerging Trends*. Ed. by M.B. Krishna. Taylor & Francis Group, 2018. Chap. 3.

[42] A. Zugenmaier. "The Freiburg privacy diamond". In: *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE* 3 (Dec. 2003), 1501–1505 vol.3. DOI: 10.1109/GLOCOM.2003.1258488.

[43] Rui Ferreira and Rui Aguiar. "Breaching location privacy in XMPP based messaging". In: *IEEE GLOBECOM*. 2012.

[44] Rui Ferreira, Alfredo Matos, and Rui Aguiar. "Hint-driven DNS resolution". In: *IEEE symposium on Computers and Communications*. ISCC'11. Corfu, Greece, 2011.

[45] *OpenDNS*. https://www.opendns.com/about/.

[46] *DNS Security with DNSCrypt*. https://www.opendns.com/about/innovations/dnscrypt/.

[47] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. *Specification for DNS over Transport Layer Security (TLS)*. RFC 7858. May 2016. DOI: 10.17487/RFC7858. URL: https://rfc-editor.org/rfc/rfc7858.txt.

[48] Rui Ferreira and Rui L. Aguiar. "Repositioning privacy concerns: Web servers controlling URL metadata". In: *Journal of Information Security and Applications* 46 (2019), pp. 121–137. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2019.03.010.

[49] Y. Rekhter and T. Li. *An Architecture for IP Address Allocation with CIDR*. RFC 1518. Internet Engineering Task Force, Sept. 1993, p. 27. URL: http://www.rfc-editor.org/rfc/rfc1518.txt.

[50] Lixia Zhang, Kevin Fall, and David Meyer. *Report from the IAB Workshop on Routing and Addressing*. RFC 4984. Sept. 2007. DOI: 10.17487/RFC4984. URL: https://rfc-editor.org/rfc/rfc4984.txt.

[51] C. Perkins. *IP Mobility Support for IPv4, Revised*. RFC 5944 (Proposed Standard). Internet Engineering Task Force, Nov. 2010. URL: http://www.ietf.org/rfc/rfc5944.txt.

[52] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. *Proxy Mobile IPv6*. RFC 5213 (Proposed Standard). Updated by RFC 6543. Internet Engineering Task Force, Aug. 2008. URL: http://www.ietf.org/rfc/rfc5213.txt.

[53] John F. Shoch. *A Note on Inter-Network Naming, Addressing, and Routing*. Internet Experiment Note #19. Jan. 1978. URL: https://www.rfc-editor.org/ien/ien19.txt.

[54] J. H. Saltzer. *On the Naming and Binding of Network Destinations*. RFC 1498. Internet Engineering Task Force, Aug. 1993, p. 10. URL: http://www.rfc-editor.org/rfc/rfc1498.txt.

[55] T. Berners-Lee, R. Fielding, and L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. Internet Engineering Task Force, Jan. 2005. URL: http://www.rfc-editor.org/rfc/rfc3986.txt.

[56] Holger Karl, Thorsten Biermann, and Hagen Woesner. "Naming and Addressing". In: *Architecture and Design for the Future Internet: 4WARD Project*. Ed. by M. Luis Correia, Henrik Abramowicz, Martin Johnsson, and Klaus Wünstel. Dordrecht: Springer Netherlands, 2011, pp. 89–113. DOI: 10.1007/978-90-481-9346-2_5.

[57] P. Nikander, J. Laganier, and F. Dupont. *An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)*. RFC 4843. Apr. 2007.

[58] John Day. *Patterns in Network Architecture: A Return to Fundamentals*. Pearson Education, 2008.

[59] S. Farrell, D. Kutscher, C. Dannewitz, B. Ohlman, A. Keranen, and P. Hallam-Baker. *Naming Things with Hashes*. RFC 6920 (Proposed Standard). Internet Engineering Task Force, Apr. 2013. URL: http://www.ietf.org/rfc/rfc6920.txt.

[60] Zooko Wilcox-O'Hearn. *Names: Decentralized, Secure, Human-Meaningful: Choose Two*. URL: http://web.archive.org/web/20111227083803/http://zooko.com/distnames.html.

[61] Daniel Shawcross Wilkerson. "A Proposal for Proquints: Identifiers that are Readable, Spellable, and Pronounceable". In: *CoRR* abs/0901.4016 (2009). URL: http://arxiv.org/abs/0901.4016.

[62] M. Stiegler. *An Introduction to Petname Systems*. Available from http://www.skyhunter.com/marcs/petnames/IntroPetNames.html. June 2010. URL: http://www.skyhunter.com/marcs/petnames/IntroPetNames.html.

[63] Harry Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, and Arvind Narayanan. "An empirical study of Namecoin and lessons for decentralized namespace design". In: *WEIS '15: Proceedings of the 14th Workshop on the Economics of Information Security* (June 2015).

[64] Ian Grigg and Philipp Güring. *Bitcoin & Gresham's Law - the economic inevitability of Collapse*. 2011. URL: http://iang.org/papers/BitcoinBreachesGreshamsLaw.pdf.

[65] Ben Laurie. *Decentralised currencies are probably impossible (but let's at least make them efficient)*. 2011. URL: http://w.fipr.org/files/decentralised-currencies.pdf.

[66] Jeremy Rand. "Case Study: Alternate Blockchains". In: *QCon Software Development Conference*. London, United Kingdom, June 2017.

[67] ITU-T. *Draft Recommendation Y.FNid Framework of identifiers in future networks*. 2012.

[68] ITU-T. *Recommendation Y.3031 Identification framework in future networks*. 2012.

[69] ITU-T. *Recommendation E.164, The International Public Telecommunication Numbering Plan*. 2005.

[70] Robert J. Stroud. "Naming Issues in the Design of Transparently Distributed Operating Systems". PhD thesis. The University of Newcastle, 1987.

[71] Juan Benet. "IPFS - Content Addressed, Versioned, P2P File System". In: *CoRR* abs/1407.3561 (2014). arXiv: 1407.3561. URL: http://arxiv.org/abs/1407.3561.

[72] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and X. Li. *IPv6 Addressing of IPv4/IPv6 Translators*. RFC 6052. Oct. 2010.

[73] M. Bagnulo, P. Matthews, and I. van Beijnum. *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers*. RFC 6146. Apr. 2011.

[74] G. Tsirtsis and P. Srisuresh. *Network Address Translation - Protocol Translation (NAT-PT)*. RFC 2766. Internet Engineering Task Force, Feb. 2000, p. 21. URL: http://www.rfc-editor.org/rfc/rfc2766.txt.

[75] Xing Li, Congxiao Bao, Wojciech Dec, Ole Troan, Satoru Matsushima, and Tetsuya Murakami. *Mapping of Address and Port using Translation (MAP-T)*. RFC 7599. July 2015. DOI: 10.17487/RFC7599. URL: https://rfc-editor.org/rfc/rfc7599.txt.

[76] Cedric Aoun and Elwyn B. Davies. *Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status*. RFC 4966. July 2007. DOI: 10.17487/RFC4966. URL: https://rfc-editor.org/rfc/rfc4966.txt.

[77] Robert E. Gilligan and Erik Nordmark. *Basic Transition Mechanisms for IPv6 Hosts and Routers*. RFC 4213. Oct. 2005. DOI: 10.17487/RFC4213. URL: https://rfc-editor.org/rfc/rfc4213.txt.

[78] M. Rambold, H. Kasinger, F. Lautenbacher, and B. Bauer. "Towards Autonomic Service Discovery A Survey and Comparison". In: *2009 IEEE International Conference on Services Computing*. Sept. 2009, pp. 192–201. DOI: 10.1109/SCC.2009.59.

[79] Bluetooth SIG. *Core Specification 4.0*. Tech. rep. Bluetooth SIG, June 2010. URL: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[80] *iBeacon, Apple Developer*. URL: https://developer.apple.com/ibeacon/.

[81] *The Physical Web*. URL: https://google.github.io/physical-web/.

[82] Shivaun Albright, Paul J. Leach, Ye Gu, Yaron Y. Goland, and Ting Cai. *Simple Service Discovery Protocol/1.0*. Internet-Draft draft-cai-ssdp-v1-03. Work in Progress. Internet Engi-

neering Task Force, Nov. 1999. URL: https://datatracker.ietf.org/doc/html/draft-cai-ssdp-v1-03.

[83] Stuart Cheshire and Marc Krochmal. *Multicast DNS*. RFC 6762. Feb. 2013. DOI: 10.17487/RFC6762. URL: https://rfc-editor.org/rfc/rfc6762.txt.

[84] Stuart Cheshire and Marc Krochmal. *DNS-Based Service Discovery*. RFC 6763. Feb. 2013. DOI: 10.17487/RFC6763. URL: https://rfc-editor.org/rfc/rfc6763.txt.

[85] S. Cheshire, B. Aboba, and E. Guttman. *Dynamic Configuration of IPv4 Link-Local Addresses*. RFC 3927. Internet Engineering Task Force, Mar. 2005. URL: http://www.rfc-editor.org/rfc/rfc3927.txt.

[86] Wifi Alliance. *Wifi direct specifications*. http://www.wi-fi.org/discover-and-learn/wi-fi-direct.

[87] *Information capacity and versions of QR Code*. 2018. URL: http://www.qrcode.com/en/about/version.html.

[88] European Payments Council. *EPC069-12 Quick Response Code: Guidelines to Enable Data Capture for the Initiation of a SEPA Credit Transfer*. Tech. rep. July 2015.

[89] S. Dey, S. Agarwal, and A. Nath. "Confidential Encrypted Data Hiding and Retrieval Using QR Authentication System". In: *2013 International Conference on Communication Systems and Network Technologies*. Apr. 2013, pp. 512–517. DOI: 10.1109/CSNT.2013.112.

[90] *Standard ECMA-340 Near Field Communication Interface and Protocol*. June 2013. URL: https://www.ecma-international.org/publications/standards/Ecma-340.htm.

[91] *Standard ECMA-385 NFC-SEC: NFCIP-1 Security Services and Protocol*. June 2015. URL: https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-385.pdf.

[92] *NFC Record Type Definition (RTD), NFC Forum Technical Specification*. July 2006. URL: https://nfc-forum.org/our-work/specifications-and-application-documents/specifications/record-type-definition-technical-specifications/.

[93] Rajalakshmi Nandakumar, Krishna Kant Chintalapudi, Venkat Padmanabhan, and Ramarathnam Venkatesan. "Dhwani: Secure Peer-to-peer Acoustic NFC". In: *SIGCOMM Comput. Commun. Rev.* 43.4 (Aug. 2013), pp. 63–74. DOI: 10.1145/2534169.2486037.

[94] M. Roland, J. Langer, and J. Scharinger. "Security Vulnerabilities of the NDEF Signature Record Type". In: *2011 Third International Workshop on Near Field Communication*. Feb. 2011, pp. 65–70. DOI: 10.1109/NFC.2011.9.

[95] Dick Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. Oct. 2012. DOI: 10.17487/RFC6749. URL: https://rfc-editor.org/rfc/rfc6749.txt.

[96] David Recordon, Johnny Bufu, Josh Hoyt, Brad Fitzpatrick, and Dick Hardt. *OpenID Authentication 2.0*. Dec. 2007. URL: http://openid.net/specs/openid-authentication-2_0.html.

[97] P. Jones, G. Salgueiro, M. Jones, and J. Smarr. *WebFinger*. RFC 7033 (Proposed Standard). Internet Engineering Task Force, Sept. 2013. URL: http://www.ietf.org/rfc/rfc7033.txt.

[98] Blaine Cook and Eran Hammer-Lahav. *Web Host Metadata*. RFC 6415. Oct. 2011. DOI: 10.17487/RFC6415. URL: https://rfc-editor.org/rfc/rfc6415.txt.

[99] P. V. Mockapetris. *Domain names: Concepts and facilities*. RFC 882. Internet Engineering Task Force, Nov. 1983, p. 31. URL: http://www.rfc-editor.org/rfc/rfc882.txt.

[100] P. V. Mockapetris. *Domain names: Implementation specification*. RFC 883. Internet Engineering Task Force, Nov. 1983, p. 73. URL: http://www.rfc-editor.org/rfc/rfc883.txt.

[101] Paul Vixie. “What DNS Is Not”. In: *Queue* 7.10 (2009), pp. 10–15.

[102] *Unbound, a validating, recursive, and caching DNS resolver*. URL: http://www.unbound.net/.

[103] D. Atkins and R. Austein. *Threat Analysis of the Domain Name System (DNS)*. RFC 3833. Internet Engineering Task Force, Aug. 2004. URL: http://www.rfc-editor.org/rfc/rfc3833.txt.

[104] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. *DNS Security Introduction and Requirements*. RFC 4033 (Proposed Standard). Updated by RFCs 6014, 6840. Internet Engineering Task Force, Mar. 2005. URL: http://www.ietf.org/rfc/rfc4033.txt.

[105] Srinivas Krishnan and Fabian Monrose. “DNS Prefetching and Its Privacy Implications: When Good Things Go Bad”. In: *Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*. LEET’10. San Jose, California: USENIX Association, 2010, pp. 10–10.

[106] *Please disable ’Perform DNS lookups to check if URLs are valid?’ by default*. 2017. URL: https://gitlab.com/gnachman/iterm2/issues/6050.

[107] Stéphane Bortzmeyer. *DNS Query Name Minimisation to Improve Privacy*. RFC 7816. Mar. 2016. DOI: 10.17487/RFC7816. URL: https://rfc-editor.org/rfc/rfc7816.txt.

[108] Liang Zhu, John Heidemann, Duane Wessels, Paul E. Hoffman, Allison Mankin, and Zi Hu. *Specification for DNS over TLS*. https://tools.ietf.org/html/draft-ietf-dprive-dns-over-tls-09. Internet-Draft. Internet Engineering Task Force, Mar. 2016.

[109] Paul E. Hoffman and Patrick McManus. *DNS Queries over HTTPS*. Internet-Draft draft-ietf-doh-dns-over-https-03. Work in Progress. Internet Engineering Task Force, Feb. 2018. 16 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-doh-dns-over-https-03.

[110] S. Sun, L. Lannom, and B. Boesch. *Handle System Overview*. RFC 3650. Internet Engineering Task Force, Nov. 2003, p. 21. URL: http://www.rfc-editor.org/rfc/rfc3650.txt.

[111] S. Sun, S. Reilly, and L. Lannom. *Handle System Namespace and Service Definition*. RFC 3651. Internet Engineering Task Force, Nov. 2003, p. 41. URL: http://www.rfc-editor.org/rfc/rfc3651.txt.

[112] S. Sun, S. Reilly, L. Lannom, and J. Petrone. *Handle System Protocol (ver 2.1) Specification*. RFC 3652. Internet Engineering Task Force, Nov. 2003, p. 53. URL: http://www.rfc-editor.org/rfc/rfc3652.txt.

[113] Antony Rowstron and Peter Druschel. “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems”. In: *Middleware 2001*. Ed. by Rachid Guerraoui. Berlin, Heidelberg: Springer, 2001, pp. 329–350.

[114] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications”. In: *SIGCOMM Comput. Commun. Rev.* 31.4 (Aug. 2001), pp. 149–160. DOI: 10.1145/964723.383071.

[115] Anjali Gupta, Barbara Liskov, and Rodrigo Rodrigues. “One Hop Lookups for Peer-to-peer Overlays”. In: *Proceedings of the 9th Conference on Hot Topics in Operating Systems - Volume 9*. HOTOS’03. Lihue, Hawaii: USENIX Association, 2003, pp. 2–2.

[116] Venugopalan Ramasubramanian and Emin Gün Sirer. “Beehive: O(1)Lookup Performance for Power-law Query Distributions in Peer-to-peer Overlays”. In: *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*. NSDI’04. San Francisco, California: USENIX Association, 2004, pp. 8–8.

[117] Ingmar Baumgart. “P2PNS: A Secure Distributed Name Service for P2PSIP”. In: *Proceedings of the Sixth Annual IEEE International Conference on Pervasive Computing and Communi-*

cations (PerCom 2008), Hong Kong, China. Mar. 2008, pp. 480–485. DOI: 10.1109/PERCOM.2008.91. URL: http://doc.tm.uka.de/2008/P2PNS_2008.pdf.

[118] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. *Peer-to-Peer Computing*. Tech. rep. HP, Apr. 2002.

[119] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. "Internet Indirection Infrastructure". In: *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '02. Pittsburgh, Pennsylvania, USA: ACM, 2002, pp. 73–86. DOI: 10.1145/633025.633033.

[120] Russ Cox, Athicha Muthitacharoen, and Robert T. Morris. "Serving DNS using a Peer-to-Peer Lookup Service". In: *In IPTPS*. 2002.

[121] Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Michael Walfish. "A Layered Naming Architecture for the Internet". In: *ACM SIGCOMM 2004*. Portland, OR, Sept. 2004.

[122] Michael Walfish, Hari Balakrishnan, and Scott Shenker. "Untangling the web from DNS". In: *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*. San Francisco, California: USENIX Association, 2004, pp. 17–17.

[123] Verisign. *The Verisign Domain Name Industry Brief*. https://www.verisign.com/assets/domain-name-report-Q42017.pdf. Dec. 2017.

[124] V. Pappas, D. Massey, A. Terzis, and L. Zhang. "A Comparative Study of the DNS Design with DHT-Based Alternatives". In: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*. Apr. 2006, pp. 1–13. DOI: 10.1109/INFOCOM.2006.207.

[125] Matthias Wachs, Martin Schanzenbach, and Christian Grothoff. "A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System". In: *Cryptology and Network Security*. Ed. by Dimitris Gritzalis, Aggelos Kiayias, and Ioannis Askoxylakis. Cham: Springer International Publishing, 2014, pp. 127–142.

[126] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler. *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. http://docs.oasis-open.org/security/saml/v2.0/. Mar. 2005.

[127] Scott Cantor, Frederick Hirsch, John Kemp, Rob Philpott, and Eve Maler. *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. http://docs.oasis-open.org/security/saml/v2.0/. Mar. 2005.

[128] John Hughes, Scott Cantor, Jeff Hodges, Frederick Hirsch, Prateek Mishra, Rob Philpott, and Eve Maler. *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*. Mar. 2005. URL: http://docs.oasis-open.org/security/saml/v2.0/.

[129] *Requirements for Internet Hosts - Application and Support*. RFC 1123. Internet Engineering Task Force, Oct. 1989, p. 98. URL: http://www.rfc-editor.org/rfc/rfc1123.txt.

[130] Simon Josefsson. *The Base16, Base32, and Base64 Data Encodings*. RFC 4648. Oct. 2006. DOI: 10.17487/RFC4648. URL: https://rfc-editor.org/rfc/rfc4648.txt.

[131] Henry Thompson and David Orchard. *URNs, Namespaces and Registries*. W3C TAG. 2006. URL: http://www.w3.org/2001/tag/doc/URNsAndRegistries-50.

[132] M. Mealling and R. W. Daniel. *URI Resolution Services Necessary for URN Resolution*. RFC 2483. Internet Engineering Task Force, Jan. 1999, p. 16. URL: http://www.rfc-editor.org/rfc/rfc2483.txt.

[133] Federico Maggi, Alessandro Frossi, Stefano Zanero, Gianluca Stringhini, Brett Stone-Gross, Christopher Kruegel, and Giovanni Vigna. "Two Years of Short URLs Internet Measurement: Security Threats and Countermeasures". In: *Proceedings of the 22Nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, 2013, pp. 861–872.

[134] Collin Jackson, Adam Barth, Andrew Bortz, Weidong Shao, and Dan Boneh. "Protecting Browsers from Dns Rebinding Attacks". In: *Proceedings of the 14th ACM Conference on Computer and Communications Security*. CCS '07. Alexandria, Virginia, USA: ACM, 2007, pp. 421–431.

[135] K. Leung, F. Le Faucheur, R. van Brandenburg, B. Downey, and M. Fisher. *URI Signing for CDN Interconnection (CDNI)*. `https://tools.ietf.org/id/draft-ietf-cdni-uri-signing-04.txt`. Internet Engineering Task Force, June 2015.

[136] *Extension for Peers to Send Metadata Files*. BEP 9. 2008. URL: `http://www.bittorrent.org/beps/bep_0009.html`.

[137] Drummond Reed, Les Chasen, and William Tan. "OpenID identity discovery with XRI and XRDS". In: *IDtrust '08: Proceedings of the 7th symposium on Identity and trust on the Internet*. Gaithersburg, Maryland: ACM, 2008, pp. 19–25.

[138] Kostas Pentikousis. "Distributed Information Object Resolution". In: *ICN '09: Proceedings of the 2009 Eighth International Conference on Networks*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 360–366.

[139] *NDN, Technical Report NDN-0022*. Tech. rep. July 2014. URL: `https://named-data.net/wp-content/uploads/2014/08/ndn-tr-22-ndn-memo-naming-conventions.pdf`.

[140] N. L. M. van Adrichem and F. A. Kuipers. "Globally accessible names in named data networking". In: *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Apr. 2013, pp. 345–350. DOI: `10.1109/INFCOMW.2013.6970714`.

[141] *CIDR REPORT*. Tech. rep. May 2018. URL: `http://www.cidr-report.org/as2.0/`.

[142] Cesar Ghali, Gene Tsudik, and Christopher A. Wood. "Network Names in Content-Centric Networking". In: *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. ACM-ICN '16. Kyoto, Japan: ACM, 2016, pp. 132–141. DOI: `10.1145/2984356.2984373`.

[143] A. Afanasyev, X. Jiang, Y. Yu, J. Tan, Y. Xia, A. Mankin, and L. Zhang. "NDNS: A DNS-Like Name Service for NDN". In: *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. July 2017, pp. 1–9. DOI: `10.1109/ICCCN.2017.8038461`.

[144] *Scalable and Adaptive Internet Solutions (SAIL)*. URL: `http://www.sail-project.eu/`.

[145] Petteri Pöyhönen and Ove Strandberg. *FP7 SAIL Deliverable 3.1, The network of information: Architecture and applications*. July 2011.

[146] Gerald Kunzmann and Dirk Staehle. *FP7 SAIL Deliverable 3.2, The Network of Information: Architecture and Applications*. July 2011.

[147] Stephen Farrell, Christian Dannewitz, Borje Ohlman, and Dirk Kutscher. *URIs for Named Information*. Internet-Draft draft-farrell-ni-00. Work in Progress. Internet Engineering Task Force, Mar. 2011. 11 pp. URL: `https://datatracker.ietf.org/doc/html/draft-farrell-ni-00`.

[148] *PURSUIT Pursuing a PUB/SUB Internet*. URL: `http://www.fp7-pursuit.eu`.

[149] Nikos Fotiou, Pekka Nikander, Dirk Trossen, and George C. Polyzos. "Developing Information Networking Further: From PSIRP to PURSUIT". In: *Broadband Communications, Networks,*

and Systems. Ed. by Ioannis Tomkos, Christos J. Bouras, Georgios Ellinas, Panagiotis Demestichas, and Prasun Sinha. Berlin, Heidelberg: Springer, 2012, pp. 1–13.

[150] D. Trossen and G. Parisis. "Designing and realizing an information-centric internet". In: *IEEE Communications Magazine* 50.7 (July 2012), pp. 60–67. DOI: 10.1109/MCOM.2012.6231280.

[151] *PSIRP Publish-Subscribe Internet Routing Paradigm*. URL: http://www.psirp.org/.

[152] Konstantinos V. Katsaros, Nikos Fotiou, Xenofon Vasilakos, Christopher N. Ververidis, Christos Tsilopoulos, George Xylomenos, and George C. Polyzos. "On Inter-domain Name Resolution for Information-centric Networks". In: *Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part I*. IFIP'12. Prague, Czech Republic: Springer-Verlag, 2012, pp. 13–26. DOI: 10.1007/978-3-642-30045-5_2.

[153] Dmitrij Lagutin. "Redesigning Internet—The Packet Level Authentication Architecture". Licentiate's Thesis. Helsinki University of Technology, Department of Information and Computer Science, 2008.

[154] Ashok Anand, Fahad Dogar, Dongsu Han, Boyan Li, Hyeontaek Lim, Michel Machado, Wenfei Wu, Aditya Akella, David G. Andersen, John W. Byers, Srinivasan Seshan, and Peter Steenkiste. "XIA: An Architecture for an Evolvable and Trustworthy Internet". In: *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. HotNets-X. Cambridge, Massachusetts: ACM, 2011, 2:1–2:6. DOI: 10.1145/2070562.2070564.

[155] Dongsu Han, Ashok Anand, Fahad Dogar, Boyan Li, Hyeontaek Lim, Michel Machado, Arvind Mukundan, Wenfei Wu, Aditya Akella, David G. Andersen, John W. Byers, Srinivasan Seshan, and Peter Steenkiste. "XIA: Efficient Support for Evolvable Internetworking". In: *Proc. 9th USENIX NSDI*. San Jose, CA, Apr. 2012.

[156] Konstantinos V. Katsaros, Nikos Fotiou, Xenofon Vasilakos, Christopher N. Ververidis, Christos Tsilopoulos, George Xylomenos, and George C. Polyzos. "On Inter-Domain Name Resolution for Information-Centric Networks". In: *NETWORKING 2012*. Ed. by Robert Bestak, Lukas Kencl, Li Erran Li, Joerg Widmer, and Hao Yin. Berlin, Heidelberg: Springer, 2012, pp. 13–26.

[157] K. V. Katsaros, X. Vasilakos, T. Okwii, G. Xylomenos, G. Pavlou, and G. C. Polyzos. "On the inter-domain scalability of route-by-name Information-Centric Network Architectures". In: *2015 IFIP Networking Conference (IFIP Networking)*. May 2015, pp. 1–9. DOI: 10.1109/IFIPNetworking.2015.7145308.

[158] W. K. Chai, N. Wang, I. Psaras, G. Pavlou, C. Wang, G. Garcia de Blas, F. J. Ramon-Salguero, L. Liang, S. Spirou, A. Beben, and E. Hadjioannou. "Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services". In: *IEEE Communications Magazine* 49.3 (Mar. 2011), pp. 112–120. DOI: 10.1109/MCOM.2011.5723808.

[159] Yu Tang, Aihua Fan, Yingjie Wang, and Yuanzhe Yao. "mDHT: a multi-level-indexed DHT algorithm to extra-large-scale data retrieval on HDFS/Hadoop architecture". In: *Personal and Ubiquitous Computing* 18.8 (Dec. 2014), pp. 1835–1844. DOI: 10.1007/s00779-014-0784-1.

[160] K. V. Katsaros, L. Saino, I. Psaras, and G. Pavlou. "On information exposure through named content". In: *Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine), 2014 10th International Conference on*. Aug. 2014, pp. 152–157. DOI: 10.1109/QSHINE.2014.6928679.

[161] Edith Ngai, Börje Ohlman, Gene Tsudik, Ersin Uzun, Matthias Wählisch, and Christopher A. Wood. "Can We Make a Cake and Eat It Too? A Discussion of ICN Security and Privacy". In: *SIGCOMM Comput. Commun. Rev.* 47.1 (Jan. 2017), pp. 49–54. DOI: 10.1145/3041027.3041034.

[162] Cesar Ghali, Gene Tsudik, and Christopher A. Wood. "(The Futility of) Data Privacy in Content-Centric Networking". In: *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*. WPES '16. Vienna, Austria: ACM, 2016, pp. 143–152. DOI: 10.1145/2994620.2994639.

[163] Jeffrey Pang, Ben Greenstein, Damon McCoy, Srinivasan Seshan, and David Wetherall. "Tryst: The Case for Confidential Service Discovery". In: *HotNets VI: The Sixth Workshop on Hot Topics in Networks*. Atlanta, GA, Nov. 2007.

[164] S. Sevilla, P. Mahadevan, and J. J. Garcia-Luna-Aceves. "iDNS: Enabling information centric networking through The DNS". In: *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Apr. 2014, pp. 476–481. DOI: 10.1109/INFCOMW.2014.6849278.

[165] E. Demirors and C. Westphal. "DNS ++: A Manifest Architecture for Enhanced Content-Based Traffic Engineering". In: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. Dec. 2017, pp. 1–6. DOI: 10.1109/GLOCOM.2017.8254708.

[166] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. "Measuring HTTPS Adoption on the Web". In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1323–1338. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/felt.

[167] Roy Thomas Fielding. "REST: Architectural Styles and the Design of Network-based Software Architectures". Doctoral dissertation. University of California, Irvine, 2000.

[168] Collin Jackson, Andrew Bortz, Dan Boneh, and John C. Mitchell. "Protecting Browser State from Web Privacy Attacks". In: *Proceedings of the 15th International Conference on World Wide Web*. WWW '06. Edinburgh, Scotland: ACM, 2006, pp. 737–744.

[169] Artur Janc and Lukasz Olejnik. "Web Browser History Detection As a Real-world Privacy Threat". In: *Proceedings of the 15th European Conference on Research in Computer Security*. ESORICS'10. Athens, Greece: Springer-Verlag, 2010, pp. 215–231.

[170] Z. Weinberg, E. Y. Chen, P. R. Jayaraman, and C. Jackson. "I Still Know What You Visited Last Summer: Leaking Browsing History via User Interaction and Side Channel Attacks". In: *2011 IEEE Symposium on Security and Privacy*. May 2011, pp. 147–161. DOI: 10.1109/SP.2011.23.

[171] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. "The Web Never Forgets: Persistent Tracking Mechanisms in the Wild". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. CCS '14. Scottsdale, Arizona, USA: ACM, 2014, pp. 674–689.

[172] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. "Cookies That Give You Away: The Surveillance Implications of Web Tracking". In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15. Florence, Italy: International World Wide Web Conferences Steering Committee, 2015, pp. 289–299. DOI: 10.1145/2736277.2741679.

[173] Markus Jakobsson and Sid Stamm. "Web Camouflage: Protecting Your Clients from Browser-Sniffing Attacks". In: *IEEE Security & Privacy* 5.6 (2007), pp. 16–24.

[174] A.G. West and A.J. Aviv. "Measuring Privacy Disclosures in URL Query Strings". In: *Internet Computing, IEEE* 18.6 (Nov. 2014), pp. 52–59.

[175] S. Varjonen, T. Heer, K. Rimey, and A. Gurtov. "Secure Resolution of End-Host Identifiers for Mobile Clients". In: *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. Dec. 2011, pp. 1–6. DOI: 10.1109/GLOCOM.2011.6134003.

[176] *DLNA Networked Device Interoperability Guidelines Expanded*. https://spirespark.com/dlna/guidelines.

[177] P. Leach, M. Mealling, and R. Salz. *A Universally Unique IDentifier (UUID) URN Namespace*. RFC 4122 (Proposed Standard). Internet Engineering Task Force, July 2005. URL: http://www.ietf.org/rfc/rfc4122.txt.

[178] Pasi Eronen, Hannes Tschofenig, Hao Zhou, and Joseph A. Salowey. *Transport Layer Security (TLS) Session Resumption without Server-Side State*. RFC 5077. Jan. 2008. DOI: 10.17487/RFC5077. URL: https://rfc-editor.org/rfc/rfc5077.txt.

[179] P. V. Mockapetris. *Domain names - implementation and specification*. RFC 1035. Internet Engineering Task Force, Nov. 1987, p. 55. URL: http://www.rfc-editor.org/rfc/rfc1035.txt.

[180] *The Directory - overview of concepts, models and services*. CCITT X.500 Series Recommendations, ITU-T. 1993.

[181] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. *The first collision for full SHA-1*. Cryptology ePrint Archive, Report 2017/190. https://eprint.iacr.org/2017/190. 2017.

[182] Andrew Banks and Rahul Gupta. *MQTT Version 3.1.1*. Tech. rep. Apr. 2015.

[183] A. Delphinanto, J.J. Lukkien, A.M.J. Koonen, F. T H Den Hartog, A. J P S Madureira, I. G M M Niemegeers, and F. Selgert. "Architecture of a Bi-Directional Bluetooth-UPnP Proxy". In: *Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE*. Jan. 2007, pp. 34–38. DOI: 10.1109/CCNC.2007.14.

[184] Seong-Hoon Kim, Jeong-Seok Kang, Hong Seong Park, Daeyoung Kim, and Young-Joo Kim. "UPnP-ZigBee internetworking architecture mirroring a multi-hop ZigBee network topology". In: *Consumer Electronics, IEEE Transactions on* 55.3 (Aug. 2009), pp. 1286–1294. DOI: 10.1109/TCE.2009.5277989.

[185] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. J. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. Internet Engineering Task Force, June 1999, p. 176. URL: http://www.rfc-editor.org/rfc/rfc2616.txt.

[186] David Naccache and Jacques Stern. "Signing on a Postcard". In: *Financial Cryptography*. Ed. by Yair Frankel. Berlin, Heidelberg: Springer, 2001, pp. 121–135.

[187] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing". In: *Journal of Cryptology* 17.4 (Sept. 2004), pp. 297–319. DOI: 10.1007/s00145-004-0314-9.

[188] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and Dave Cooper. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: 10.17487/RFC5280. URL: https://rfc-editor.org/rfc/rfc5280.txt.

[189] Warwick Ford and Yuri Poeluev. *The Machine-to-Machine (M2M) Public Key Certificate Format*. Internet-Draft draft-ford-m2mcertificate-00. Work in Progress. Internet Engineering Task Force, Mar. 2015. 14 pp. URL: https://datatracker.ietf.org/doc/html/draft-ford-m2mcertificate-00.

[190] W. Ford and Y. Poeluev. *An efficient certificate format for ECC*. http://csrc.nist.gov/groups/ST/ecc-workshop-2015/presentations/session2-ford-warwick.pdf. Workshop on Elliptic Curve Cryptography Standards, June 2015.

[191] Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal de Lara. "Amigo: Proximity-Based Authentication of Mobile Devices". In: *UbiComp 2007: Ubiquitous Computing*. Ed. by John Krumm, Gregory D. Abowd, Aruna Seneviratne, and Thomas Strang. Berlin, Heidelberg: Springer, 2007, pp. 253–270.

[192] Daniel J. Solove. *Understanding privacy*. Harvard University Press, 2008.

[193] P. Saint-Andre. *XHTML-IM*. XEP 0071. XSF, Sept. 2008. URL: http://www.xmpp.org/extensions/xep-0071.html.

[194] Peter Saint-Andre, Peter Millard, Thomas Muldowney, and Julian Missig. *"User Avatar"*. XEP 0084. XSF, Aug. 2008. URL: http://www.xmpp.org/extensions/xep-0153.html.

[195] P. Saint-Andre. *vCard-Based Avatars*. XEP 0153. XSF, Sept. 2006. URL: http://www.xmpp.org/extensions/xep-0153.html.

[196] Peter Saint-Andre. *vcard-temp*. XEP 0054. XSF, July 2008. URL: http://www.xmpp.org/extensions/xep-0153.html.

[197] Peter Eckersley. "How Unique Is Your Web Browser?" In: *Privacy Enhancing Technologies*. Ed. by Mikhail J. Atallah and Nicholas J. Hopper. Vol. 6205. Lecture Notes in Computer Science. Springer, July 8, 2010, pp. 1–18.

[198] Steven Englehardt and Arvind Narayanan. "Online Tracking: A 1-million-site Measurement and Analysis". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: ACM, 2016, pp. 1388–1401. DOI: 10.1145/2976749.2978313.

[199] Marianna Rapoport, Philippe Suter, Erik Wittern, Ondrej Lhoták, and Julian Dolby. "Who you gonna call? Analyzing Web Requests in Android Applications". In: *CoRR* abs/1705.06629 (2017).

[200] Linda Naeun Lee, Richard Chow, and Al M. Rashid. "User Attitudes Towards Browsing Data Collection". In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA '17. Denver, Colorado, USA: ACM, 2017, pp. 1816–1823. DOI: 10.1145/3027063.3053078.

[201] H. Said, N. Al Mutawa, I. Al Awadhi, and M. Guimaraes. "Forensic analysis of private browsing artifacts". In: *2011 International Conference on Innovations in Information Technology*. Apr. 2011, pp. 197–202. DOI: 10.1109/INNOVATIONS.2011.5893816.

[202] Kiavash Satvat, Matthew Forshaw, Feng Hao, and Ehsan Toreini. "On the privacy of private browsing – A forensic approach". In: *Journal of Information Security and Applications* 19.1 (2014), pp. 88–100. DOI: 10.1016/j.jisa.2014.02.002.

[203] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Tor: The Second-generation Onion Router". In: *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*. SSYM'04. San Diego, CA: USENIX Association, 2004, pp. 21–21.

[204] Paul Vixie, Sue Thomson, Y. Rekhter, and Jim Bound. *Dynamic Updates in the Domain Name System (DNS UPDATE)*. RFC 2136. Internet Engineering Task Force, Apr. 1997. URL: http://www.rfc-editor.org/rfc/rfc2136.txt.

[205] Frank Wang and James Mickens. "Veil: Private Browsing Semantics Without Browser-side Assistance". In: *NDSS*. San Diego, CA, 2018.

[206] Dietwig Lowet and Daniel Goergen. "Co-browsing Dynamic Web Pages". In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: ACM, 2009, pp. 941–950. DOI: 10.1145/1526709.1526836.

[207] Gaurav Aggarwal, Elie Bursztein, Collin Jackson, and Dan Boneh. "An Analysis of Private Browsing Modes in Modern Browsers". In: *Proceedings of the 19th USENIX Conference on Security*. USENIX Security'10. Washington, DC: USENIX Association, 2010, pp. 6–6.

[208] B. Zhao and P. Liu. "Private Browsing Mode Not Really That Private: Dealing with Privacy Breach Caused by Browser Extensions". In: *2015 45th Annual IEEE/IFIP International Con-*

ference on Dependable Systems and Networks. June 2015, pp. 184–195. DOI: 10.1109/DSN.2015.18.

[209] *IST SWIFT. Secure Widespread Identity for Federated Telecommunications (SWIFT). EU FP7-2008 Contract 215832*. URL: http://www.ist-swift.eu.

[210] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari. *Client subnet in DNS queries - Draft*. RFC. Apr. 2016. URL: https://tools.ietf.org/html/draft-ietf-dnsop-edns-client-subnet-08.

[211] Rob Austein. *DNS Name Server Identifier (NSID) Option*. RFC 5001. Aug. 2007. DOI: 10.17487/RFC5001. URL: https://rfc-editor.org/rfc/rfc5001.txt.

[212] *Name Server Daemon (NSD)*. URL: http://www.nlnetlabs.nl/projects/nsd/.

[213] P. Vixie. *Extension Mechanisms for DNS (EDNS0)*. RFC 2671. Internet Engineering Task Force, Aug. 1999, p. 7. URL: http://www.rfc-editor.org/rfc/rfc2671.txt.

[214] I. Paterson, D. Smith, P. Saint-Andre, and J. Moffitt. *XEP-0070: Verifying HTTP Requests via XMPP*. Draft Standard. XMPP Standards Foundation, Dec. 2005. URL: http://xmpp.org/extensions/xep-0070.html.

[215] A. Kumar, J. B. Postel, C. Neuman, P. Danzig, and S. Miller. *Common DNS Implementation Errors and Suggested Fixes*. RFC 1536. Internet Engineering Task Force, Oct. 1993, p. 12. URL: http://www.rfc-editor.org/rfc/rfc1536.txt.

[216] Ilya Moiseenko, Mark Stapp, and David Oran. "Communication Patterns for Web Interaction in Named Data Networking". In: *Proceedings of the 1st ACM Conference on Information-Centric Networking*. ACM-ICN '14. Paris, France: ACM, 2014, pp. 87–96. DOI: 10.1145/2660129.2660152.