BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

LAWFUL INTERCEPTION: IDENTITY DETECTION

ZÁKONNÉ ODPOSLECHY: DETEKCE IDENTITY

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR — Ing. LIBOR POLČÁK
AUTOR PRÁCE

SUPERVISOR — prof. Ing. MIROSLAV ŠVÉDA, CSc.
ŠKOLITEL

BRNO 2017

Abstract

Internet has became a regular communication channel between law offenders performing malicious activities. Courts or prosecutors authorise lawful interception that targets individual suspects. Lawful interception systems have to identify traffic of the suspects. However, the identifiers that appear in each network packet are short-lived, dynamically assigned, and a single communication session may be split into multiple flows identified by different identifiers. A lawful interception system has to immediately detect that a new identifier covered by an intercept appeared or disappeared. This thesis describes identification in modern computer networks compatible with lawful interception. The focus is on two partial identity detectors: IPv6 address assignment tracking based on monitoring of neighbor discovery and clock-skew-based remote computer identification. Additionally, this thesis proposes identity graphs that link partial identities according to optional constraints that reflect the wording of a warrant. Results of partial identity linking can be utilised by lawful interception systems. The results of this thesis are also applicable in network forensic, and in networks controlled according to user roles.

Abstrakt

Komunikace předávaná skrze Internet zahrnuje komunikaci mezi pachateli těžké trestné činnosti. Státní zástupci schvalují cílené zákonné odposlechy zaměřené na podezřelé z páchání trestné činnosti. Zákonné odposlechy se v počítačových sítích potýkají s mnoha překážkami. Identifikátory obsažené v každém paketu jsou koncovým stanicím přidělovány po omezenou dobu, nebo si je koncové stanice dokonce samy generují a automaticky mění. Tato dizertační práce se zabývá identifikačními metodami v počítačových sítích se zaměřením na metody kompatibilní se zákonnými odposlechy. Zkoumané metody musejí okamžitě detekovat použití nového identifikátoru spadajícího pod některý z odposlechů. Systém pro zákonné odposlechy následně nastaví sondy pro odposlech komunikace. Tato práce se převážně zabývá dvěma zdroji identifikačních informací: sledováním mechanismu pro objevování sousedů a detekcí identity počítače na základě přesností měření času jednotlivých počítačů. V rámci dizertačního výzkumu vznikly grafy identit, které umožňují spojování identit s ohledem na znění povolení k odposlechu. Výsledky výzkumu je možné aplikovat v rámci zákonných odposlechů, síťové forenzní analýzy i ve vysokoúrovňových programově řízených sítích.

Keywords

Lawful interception, IPv6 address tracking, neighbor discovery tracking, clock-skew-based fingerprinting, partial identity linkage.

Klíčová slova

Zákonné odposlechy, detekce přiřazení adres IPv6, identifikace počítače pomocí odchylky měření času, spojování částečných identit.

Reference

POLČÁK, Libor. *Lawful Interception: Identity Detection.* Brno, 2017. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Švéda Miroslav.

Lawful Interception: Identity Detection

Declaration

Hereby I declare that this PhD thesis was prepared as an original author's work under the supervision of prof. Ing. Miroslav Švéda, CSc. with expert supervision of Ing. Petr Matoušek, PhD, M.A. During the work on this PhD thesis I managed a group of students investigating lawful interception; I cooperated with Ing. Tomáš Martínek, PhD, and students of bachelor and master degree study program. Some tools and ideas were developed as a part of this cooperation under my supervision. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

....
Libor Polčák
October 17, 2017

Acknowledgements

This PhD thesis was supported by the project VG20102015022 (Modern Tools for Detection and Mitigation of Cyber Criminality on the New Generation Internet) supported by the Ministry of the Interior of the Czech Republic. This work was also supported by The Ministry of Education, Youth and Sports of the Czech Republic from (1) the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science — LQ1602 and (2) the project TeamIT - Building Competitive Research Teams in IT — EE2.3.09.0067.

I would like to thank reviewers of my papers for interesting comments that moved my PhD research forward. The rest of the acknowledgment is written in Czech.

Děkuji mé ženě Markétě za lásku a podporu a ostatním členům rodiny za podporu při vypracování práce.

Děkuji mému školiteli Mirkovi Švédovi za konzultace k práci, zajímavé diskuze a pomoc s jazykem. Děkuji mému školiteli specialistovi Petrovi Matouškovi za vedení projektu Sec6Net, za dlouhé hodiny strávené při čtení práce, poznámky k práci a diskuze k práci. Děkuji kolegům Ondrovi Ryšavému, Matěji Grégrovi a Vladmimírovi Veselému za diskuze k práci a podnětné myšlenky. Děkuji ostatním členům skupiny NES@FIT za podporu.

Děkuji mým studentům za vypracované bakalářské a diplomové práce, především Báře Frankové, Martinu Holkovičovi, Radku Hranickému a Jakubovi Jiráskovi za přispění k mé publikační činnosti. Děkuji Tomovi Martínkovi za to, že přispěl k vedení studentů v rámci projektu Sec6Net.

Děkuji Honzovi Kořenkovi a dalším členům projektu Sec6Net za sondy, které na vysokých rychlostech využívají vytvořený nástroj systém pro zákonné odposlechy SLIS.

Contents

List of abbreviations

AF	Administration Function, page 31.
ALIS	Aqsacom real time Lawful Interception System, page 37.
API	Application interface, page 135.
BRAS	Broadband remote access server, page 179.
CC	Content of Communication, page 28.
CC-IIF	Content of Communication – Internal Interception Function, page 31.
CCC	Call Content Channel, page 32.
CCTF	Content of Communication Trigger Function, page 31.
CDC	Call Data Channel, page 32.
CGN	Carrier Grade NAT, page 42.
DAD	Duplicate Address Detection, page 25.
DHCP	Dynamic Host Configuration Protocol, page 9.
DHCPv6	Dynamic Host Configuration Protocol version 6, page 23.
DSL	Digital subscriber line, page 179.
DSLAM	DSL access multiplexer, page 179.
DUID	DHCPv6 Unique Identifier, page 23.
ETSI	European Telecommunications Standards Institute, page 27.
HE	Happy Eyeballs, page 46.
HI	Handover Interface, page 30.
IAN	Identity Aware Networks, page 8.
IAP	Intercept Access Point, page 31.
ICMPv6	Internet Control Message Protocol for the Internet Protocol Version 6, page 24.
IID	Interface Identifier, page 23.

IMS	Identity Management System, page 17.
IOI	Items of Interests, page 14.
IP	Internet Protocol, page 7.
IPv4	Internet Protocol version 4, page 8.
IPv6	Internet Protocol version 6, page 8.
IRI	Intercept Related Information, page 28.
IRI-IIF	Intercept Related Information – Internal Interception Function, page 31.
ISP	Internet Service Provider, page 7.
ISS	Intelligence Support Systems, page 38.
IXP	Internet Exchange Point, page 42.
LAN	Local Area Network, page 7.
LEA	Law Enforcement Agency, page 27.
LEMF	Law Enforcement Monitoring Facility, page 27.
LI	Lawful Interception, page 7.
MF	Mediation Function, page 31.
MLD	Multicast Listener Discovery, page 24.
NA	Neighbor Advertisement, page 25.
NAI	Network Access Identifier, page 30.
NAT	Network Address Translation, page 8.
NC	Neighbor Cache, page 53.
ND	Neighbor Discovery, page 23.
NS	Neighbor Solicitation, page 25.
NS-DAD	Neighbor Solicitation issued during DAD, page 25.
OF	OpenFlow, page 134.
OSINT	Open Source Investigation, page 137.
pcf	PC Fingerprinter, page 77.
RA	Router Advertisment, page 24.
RS	Router Solicitation, page 24.
SAS	Surveillance Administration System, page 32.

SDN	Software-defined networking, page 9.
Sec6Net	Modern Tools for Detection and Mitigation of Cyber Criminality on the New Generation Internet, page 9.
SIMS	Sec6Net Identity Management System, page 9.
SLAAC	Stateless Address Autoconfiguration, page 23.
SLIS	Sec6Net Lawful Interception System, page 9.
SProbe	Smart Application Aware Embedded Probes, page 47.
TSP	Telecommunication Service Provider, page 33.
URI	Unique resource identifier, page 20.
VoIP	Voice over IP, page 35.
WAN	Wide Area Network, page 8.

Chapter 1

Introduction

Throughout the evolution of humanity, the identity of a particular person was strongly connected to its physical presence and appearance [162]. Each individual was typically identified by their physical features, voice and similar characteristics. Later, authorities introduced unique identifiers, such as passport numbers and personal ID numbers.

With the advent of information technologies, so-called digital and virtual identities emerged. As people employ digital and virtual identities remotely, these are not connected to the appearance. Typically, visitors are not required to validate their identity by official identifiers when they access services offered on the Internet. As a result, people create and manage a score of digital identities; some of them are connected to activities such as work, leisure, and social networking. Some of these identities are only loosely connected, and it is not obvious to a remote observer that all belong to the same person.

Computer networks present many novel means for communication of individuals. For example, people can communicate in the form of e-mail [164], instant messaging (e.g., XMPP [169]), or Voice over IP services (e.g., SIP [167]). Besides these open protocols, many service providers employ proprietary protocols. The recent arrival of mobile applications increases the number of new services providing communication between individuals.

Not only are the computer networks used for benign activities but also malicious users, criminals, and terrorists use the same networks and protocols [26, 87, 90, 115]. To mitigate these threats, standards for *lawful interception* (LI), originally developed for telephone networks, were transformed for computer networks [9, 55]. In the European Union, the European Council allowed LI in 1996 [179]. Later, the Council of Europe approved the *Convention on Cybercrime* [41]. The Czech Republic adopted the European law [118].

The aim of LI is to gather complete and indisputable evidence of criminal activities. In telephone networks, the subject of an interception (the intercept target) can be identified by his or her telephone number; both public switched telephone network operators and mobile carriers can accurately identify the subject and the data related to the intercept.

In computer networks, identification of an LI target is more complicated [44]. It is necessary to distinguish the traffic of the intercept target from the traffic of other users of the network. Typically, there is not a static unique identifier for each person that is available across networks.

A MAC address identifies a network interface of a computer on a *local area network* (LAN), the interception in the network of the *internet service provider* (ISP) has to rely on other identifiers.

Nowadays, *Internet Protocol* (IP) is the only protocol commonly used on the Internet to address hosts across network boundaries. Each computer carries one or more IP addresses.

Since 1983, the dominant version of IP has been 4 (IPv4) [98]. However, its address space is limited and insufficient for today requirements [168]. The inevitable exhaustion of the IPv4 address space was evident in the early 1990s. Consequently, efforts to conserve the address space emerged. *network address translation* (NAT) hides a LAN [163] or a *wide area network* (WAN) [193] behind a limited number of IPv4 addresses. Small offices and home networks usually get one public (globally unique) IPv4 address each. The deployment of the IPv4 successor — IP version 6 (IPv6) [45] is currently in the process [48]. Each IPv6 host can simultaneously use as many addresses as it can handle [135]. Hence, IPv6 addresses are dynamically configured, short-lived and often random.

This thesis investigates methods for computer and person identification in modern networks. The focus is on methods applicable in LI.

Nevertheless, the technical aspects are only one side of the coin. On the other, there are the ethical principles and the right to privacy. The Constitutional Order of the Czech Republic [39] acknowledges the right to privacy. Most of the other countries have similar guarantees. The contrast between the respect for privacy and the need to counter serious crime and terrorists raised a worldwide controversy [64, 190]. Although this thesis primarily focuses on technical aspects of LI, it also emphasises privacy-related questions, especially in the Chapter 4.

1.1 Research objectives

This PhD research is divided into the following areas:

1. Overview of LI standards and state-of-the-art of the identification in computer networks — during the research, we identified several challenges arising in modern computer networks [146, 157].
2. Methods for local and remote identification with the main focus on methods for identification in IPv6 networks — during the research, we proposed the IPv6 address assignment tracking deployable on IPv6 LANs [154, 156] and studied clock-skew-based remote computer identification and its applications [147, 148, 158].
3. Methods that link identity-related information of a single person or a computer together — we proposed identity graphs, a formal mechanism that links discovered identities that belong to a single subject [157].
4. Applications of the mechanisms studied in the research topics mentioned above — the methods are applicable in LI [157, 159] and *Identity Aware Networks* (IAN) [160].

The following research hypotheses arise for the second and third part of this PhD research:

Hypothesis 1. *The detection of IPv6 address assignments is possible from Neighbor Discovery traffic [136] even in networks with MLD snooping enabled (neighbor discovery traffic is multicasted rather than broadcasted).*

Hypothesis 2. *Short-term clock skew estimates can uniquely identify computers for LI.*

Hypothesis 3. *Identity information revealed at different locations can be linked by applying unambiguous rules.*

1.2 Achieved results

This PhD research was a part of the research project VG20102015022 *Modern Tools for Detection and Mitigation of Cyber Criminality on the New Generation Internet* (Sec6Net)[1] supported by the Ministry of the Interior of the Czech Republic. Based on the study of related work, challenges in modern and future networks were identified and presented at ISS Europe[2] 2013; and later published [146, 157]. The methods for identification were divided into two groups: local and remote.

Local methods are applicable on LANs. Usually, they monitor local traffic or require access to nodes deployed on the LAN. IPv6 provides a new decentralised mechanism that enables hosts to generate IPv6 addresses without any authority, such as *Dynamic Host Configuration Protocol* (DHCP) server. As a part of this PhD research, we analysed the behaviour of different operating systems and proposed IPv6 address assignment tracker deployable on LANs [91, 154, 156]. The IPv6 address assignment tracking translates IPv6 address management traffic to a sequence of symbols announcing the activity and inactivity of an IPv6 address. The output of the timed transducer is suitable for generating messages for LI. The proposed tracking fulfils Hypothesis 1.

Remote identification methods are usually less accurate than local methods. Nevertheless, their benefit is that they do not depend on the information available only in the network of the subject to be identified. Clock-skew-based identification [112] seemed to be a powerful tool for remote identification. However, its evaluation revealed limits [69, 104, 147, 158] of its applicability. Consequently, Hypothesis 2 was rejected.

As a part of the Sec6Net research project, other identification methods were studied, either by me or under my supervision. As a result, several modules for identity detection were developed [103, 159, 173]. These modules became a basis for a tool called *Sec6Net Identity Management System* (SIMS) [151]. SIMS gathers information about identities from several partial identity detectors and links them together.

The linkage of identities is based on the study of the identifiers used in computer networks and their relations. Based on this study, identity graphs of network identifiers were proposed [157] and implemented [95]. Identity graphs can link different identifiers based on conditions specified by a warrant for LI. This thesis extends the original identity graphs with more operations, including time-related operations and support for inaccurate partial identity detectors. Identity graphs confirm Hypothesis 3.

Identity graphs were applied in the LI system called *Sec6Net Lawful Interception System* (SLIS) [152] developed as a part of the Sec6Net project. In addition, *software-defined networking* (SDN) proved to be useful [92, 160] for prototyping other applications benefiting from the identification mechanisms developed during this PhD research.

1.3 Organisation of this thesis

This thesis is divided into three parts; each is further divided into chapters. The first part, *Essential background*, presents terminology and necessary background for this thesis and elaborates on challenges related to the identification in modern and future computer networks. The following part, *Identity detection and linkage*, describes the main results of this PhD research. The main goal of this part is to provide information that allows to

[1] https://www.fit.vutbr.cz/~ipolcak/grants.php?id=517

[2] Intelligence Support Systems for Lawful Interception, Electronic Surveillance and Cyber Intelligence Gathering, http://www.issworldtraining.com/ISS_EUROPE/

accept or reject the three hypotheses. The final part, *Applicability of the results of this thesis*, provides application of results of this PhD research and concludes the thesis.

Part I introduces the necessary background to identification in LI in modern and future computer networks. It is organised as follows:

- Chapter 2 summarises the identity from the theoretical point of view and defines the identity-related terminology used in this thesis.
- Chapter 3 covers basics of networking that are necessary to understand this thesis, including network-related identifiers. Section 3.3 focuses on basics of IPv6 address assignment. This chapter also explains the difference between local and remote identification.
- Chapter 4 focuses on LI from both legal and technical view. Section 4.1 overviews European LI standards and compares them to other views on LI around the world. Section 4.2 presents related work to the research in the area of LI. Section 4.3 elaborates on interests of the stakeholders in the area of LI.
- Chapter 5 provides five challenges that this PhD research identified in the LI in modern networks.

The second part, *Identity detection and linkage*, presents the main results of this thesis.

- Chapter 6 focuses on Hypothesis 1 and computer identification in local IPv6 networks. It provides the related work in the IPv6 identification and a study of IPv6 address assignment implementation in different operating systems. The study provides necessary background to define the IPv6 address tracking based on the timed transducer that detects active IPv6 addresses. The implementation of the tracking is evaluated in laboratory and real IPv6 network. Based on this chapter, Hypothesis 1 is accepted.
- Chapter 7 presents the properties of remote computer identification based on observation of clock skew inbuilt to each clock due to inevitable differences on the atomic level. The aim is the evaluation of Hypothesis 2. It is possible to link the addresses even from data intercepted from a remote location. The chapter elaborates on requirements for correct clock skew estimation both formally and empirically. Chapter 7 also lists possible challenges associated with the clock-skew-based identification, for example, a possibility to mimic clock skew of another computer and a real network deployment. Observed results show that Hypothesis 2 can be rejected.
- Chapter 8 presents the study of identifiers in computer networks with the aim at Hypothesis 3. This chapter defines identity graphs, a formal model based on graph operations that allow linkage of partial identities. Time-related identity linkage with respect to the nuances in the wording of LI warrants is possible. The model supports linkage of identities learnt from partial identity detectors of limited accuracy. The proposed model confirms Hypothesis 3.

The final part, *Applicability of the results of this thesis*, provides applications for the identity detection discussed in this thesis. Part III is organised as follows:

- Chapter 9 overviews practical deployment of the methods discussed in this thesis. The methods were implemented as a part of SLIS. Chapter 9 also describes high-level SDN, which allows rules containing user identifiers. Identity graphs link such identifiers to IP addresses or flow identifiers supported by SDN.

- Chapter 10 summarises and concludes this thesis. In addition, this chapter also proposes future work. Chapter 10 lists publications produced during this PhD research and describes their relation to this thesis.

Part I

Essential background

Chapter 2

Theory of Identity

People used to live in small societies spanning a village and its surrounding or a part of a town. Most people used to have only one identity [162] during the history of mankind as they were known by their neighbours and fellows. People usually did not have means to hide their identity and often lacked the motivation to do so. However, the advent of the information society and Internet changed the balance. So-called digital and virtual identities enabled the general public [162] to split identity for different Internet activities, such as related to work, families, hobbies.

This chapter focuses on the description of technical aspects of an identity from the theoretical point of view. This chapter defines identity based on the terminology introduced by other researchers and related papers. This thesis uses the terminology established in this chapter.

2.1 Digital identities

Pfitzmann together with fellow scientists from the field of identity management, anonymity, pseudonymity, and unobservability developed the terminology[1] [144, 145] connected to identities and anonymity based on previous papers of the aforementioned fields [27, 34, 85]. The terminology is based on *entities* (subjects and objects) and actions. In their terminology, subjects execute actions on objects, for example, a subject sends a message.

Firstly, let us focus on the *subjects*. As Figure 2.1 shows, the network environment encloses different types of *actors*. There are human beings possibly operating several devices (for example, a cell phone, a notebook, a desktop, a tablet), some of which may communicate through more than one connection with the network (for example, Wi-Fi and cellular network). Additionally, there are software tools called bots crawling the network [175]. Typically, their goal is to create a database containing information from various websites, for example, bots operated by search engines index information contained on the web, bots looking for bargain offers in e-shops and auction servers [73]). A group (consisting of several people or bots) can represent an organisation or a legal person. People move from one place to another. They spent a part of a day at home. Then they move to work. Some people can connect to a network while they commute or from a café or a restaurant they visit. Some devices (usually mobile devices) can be used both in work and while people focus on their

[1] A note for Czech speaking readers: the report [144] also contains the Czech translation of the terminology.

personal life. Other devices are strictly bound to a specific role, for example, a desktop at work.

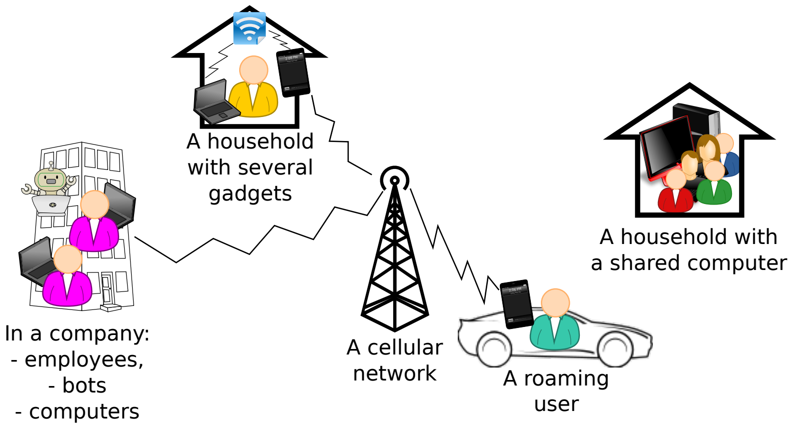


Figure 2.1: There are several subjects that can be identified in the network environment, including human beings, bots, computers and other gadgets.

Pfitzmann and Hansen [144] regard actors and actees as a subject of identification if they are human beings, legal persons, or a computer. They distinguish between a subject and a set of subjects for the study of pseudonyms and group pseudonyms.

Pfitzmann and Hansen [144] consider *systems* with the following properties:

1. A system has a surrounding which is a part of the world outside of the system. The system and its surrounding form the universe.

2. The state of the system may change by actions within the system.

Figure 2.2 shows a system and its surrounding. The goal of an adversary is to identify the subjects — the senders and the receivers of the messages (objects) within the system. Senders and receivers located in the surrounding are not subjects of identification. Pfitzmann and Hansen consider entities to be a set of all subjects and objects in the system.

Items of Interests (IOI) [144] is any of the following: (a) a subject, (b) any characteristic of a subject, (c) an action performed by a subject, or (d) a message exchanged by some subjects and (e) properties of such a message. IOI can be a specific property, for example, a *home address*. Other IOIs refer to actions that happen in the system, such as *the person A sent a message X*.

Attributes are defined as "a quality or characteristic of an entity or an action" [144]. Hence, an attribute is either a member of the set of all IOIs, or the observation of attributes may reveal some IOIs — actions within the system.

An *Attribute value* is not only the value of an attribute, but the term also refers to the type of the attribute itself [144]. Therefore, the pair *("location", "Božetěchova 2, Brno, the Czech Republic")* is an attribute value while the address *Božetěchova 2, Brno, the Czech Republic* is not an attribute value on its own.

Attribute values can be either static (permanent, typically in-born) or dynamic (temporary) [85, 132].

Finally, Pfitzmann and Hansen define an identity as "any subset of attribute values of an individual person which sufficiently distinguishes this individual person from all other persons within any set of persons" [144].

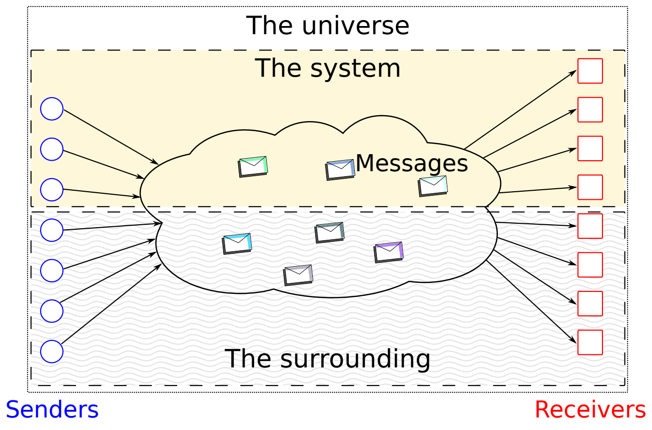


Figure 2.2: The system with subjects (senders and receivers) to be identified by an adversary. An adversary identify senders and receivers in the system only.

Therefore, the goal of the methods for identification in the model depicted in Figure 2.2 is to provide enough attribute values to distinguish all subjects (senders and receivers) in the system.

Pfitzmann and Hansen [144] also allow probabilistic identification. During probabilistic identification, the identification method is not able to indisputably reveal a set of unique attribute values. Nevertheless, the method reveals the identity with some degree of certainty.

A single identity may comprise of different *partial identities* of the same person or a computer [144]; each of the partial identity represents the subject in a specific context or a role. All partial identities co-exist and can be *linked* (correlated). If two identities are *linkable*, their attributes can be merged as all belong to the same subject. Consider a subject that owns two e-mail addresses — x any y. Both x and y are partial identities of the same subject.

In the network environment, often, one of the attributes is unique to a specific identity. Such attribute is called the *identifier* — usually, it exists in the form of a name or a bit string [144]. Each identifier corresponds to a specific (partial) identity. For instance, the identity of an e-mail user owning a mailbox of an address x can be represented directly by the identifier x.

If a group of subjects is not identifiable within the system, the set of indistinguishable subjects is called the *anonymity set* and the subjects are *anonymous* within the anonymity set.

Pfitzmann and Hansen [144] define the *anonymity* of a subject from the perspective of an adversary as the state in which the adversary "cannot sufficiently identify the subject within a set of subjects, the anonymity set". Therefore, they underline the possibility to quantify the anonymity and the need to set the threshold where anonymity starts.

Both *digital and virtual identities* refer to identities observable in the cyber space — in computer systems and networks. The main distinction is that digital identity refers directly to a specific person and his or her actions that happen in the cyber space [144]. Digital identity is directly connected to the human being and represents the attribute values and actions of the human being in the cyber space. On the other hand, virtual identity is an identity of a character in a virtual game [144]. Therefore, the actions of a virtual character with a virtual identity do not necessary mean that the person that controls the virtual character is willing to perform the same actions in the real world. However, the definition of digital and virtual identities is blurred as some literature, for example, the final report of the European FIDIS project [162], does not distinguish between digital and virtual identities.

Nevertheless, the view of Pfitzmann and Hansen is not the only one that occurs in the literature. Modinis study [176] for the eGovernment Unit of European Commission lists a comparable definition for identity and identifiers.

Identity-related terms like identity and linkability are also defined in Common Criteria for Information Technology Security Evaluation (ISO ISO/IEC 15408) [36, 100]. However, due to the scope of the standard, the definitions focus on human actors only.

Jøsang et al. [105] present a slightly different terminology depicted in Figure 2.3. In their terminology, each physical person or an organisation can have several identities. Each identity is tied to several characteristics. Characteristic elements used for identification are called identifiers. Jøsang et al. explain that identifiers are usually unique in a specific context only.

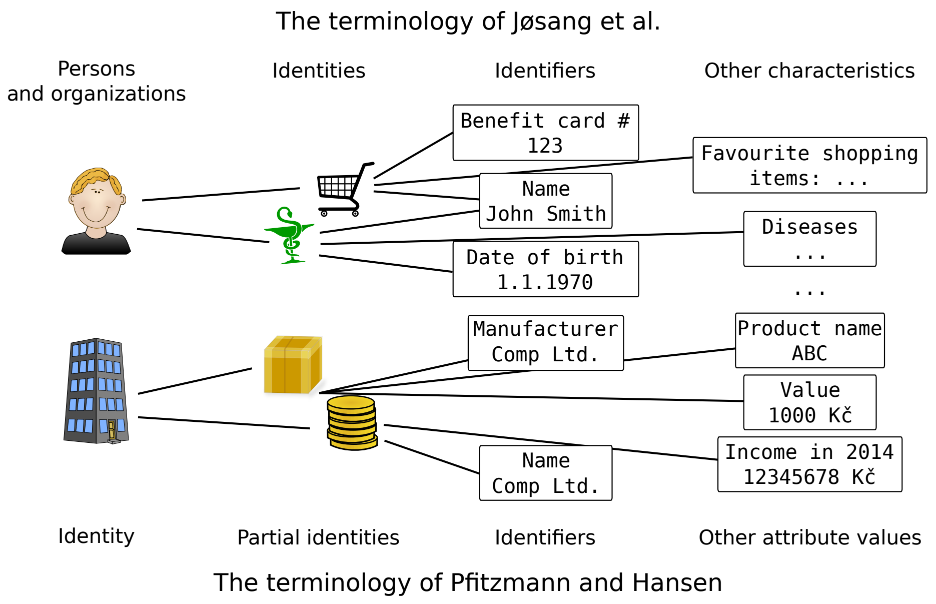


Figure 2.3: Comparison of the alternative terminology used by Jøsang et al. [105] and the equivalent terms in the terminology defined by Pfitzmann and Hansen [144].

As Figure 2.3 shows, the terminology of Jøsang et al. can be translated to the terminology of Pfitzmann and Hansen and vice versa. Although the meaning of terms used by Jøsang et al. is shifted in comparison to the terminology of Pfitzmann and Hansen, it is clear that Pfitzmann and Hansen did not omit any substantial detail. This thesis adopted the terminology of Pfitzmann and Hansen.

Nabeth [132] argues that two perspectives of the concept of an identity exist:

1. Identity as a representation – a structural perspective. The identity is represented as "a set of attribute values[2] characterising the person".

[2]The term attributes used in the original definition was replaced by the term attribute values to be coherent with the terminology of Pfitzmann and Hansen [144] adopted by this thesis.

2. Identity for identification – a process perspective. From this perspective, "identity is considered according to a set of processes relating to disclosure of information about the person and usage of this information."

The former definition is compatible with the one given by Pfitzmann and Hansen. The structural perspective covers partial identities, different contexts and the set of attribute values related to the identity. Informally, structural perspective is interested in the description of the subject and in learning its qualities and other types of IOI.

In contrast, the process perspective refers to the act of the identification. This perspective inspects the elements from which the identity can be learnt, such as names, photographs, fingerprints, or genetic patterns. Informally, the objective of the process perspective is to identify a person and learn his or her identifiers.

2.2 Identity management systems

After digital identities emerged, people started to create partial identities. These identities can be managed by their owners, corporations (for example, for marketing), and also by adversaries that illegally collects digital identities and related attribute values and IOIs.

FIDIS, a European project that focused on the identification, generalises *Identity Management System* (IMS) as "technical system supporting the process of management of (partial) identities" [123, page 131]. Such a system typically performs at least one of the following activities [123, page 131]:

- Searching for the real person operating the partial identities and linking these identities together.
- Identification, authentication, authorization, accounting, access control of both IT resources and real resources (for example, estates, buildings).
- Management of the accounts related to the identity of physical persons, for example, creating accounts for employees, assignment of privileges, deactivation.
- Aggregation and linkage of attributes, for example, for advertisement. The attributes may belong either to individuals or a group of persons (aggregated attributes).
- Application of anonymisation and pseudonymisation techniques.
- Selection of specific partial identities in a predefined context.

FIDIS also developed [123, Section 4.1] two classifications of IMSes described below.

Classification of identity management systems according to the aspect of control
Some IMSes are used directly by the person that wants to manage his or hers identity. Other IMSes are operated by organisations and manage (partial) identities of persons in relation with the organization, such as employees or customers. According to the data control, there exist three *types* of IMSes [123, page 131]:

1. IMSes for central account management, authentication, authorization, and accounting. Type 1 IMSes assign temporary characteristics such as a job title or a phone number.

2. IMSes for profiling of user data by an organisation for marketing or provisioning of personalised services. Type 2 IMSes provide an abstracted or aggregated identity.

3. IMSes for pseudonym management. These systems are controlled by the end users (they are user-centric). The user controls by the Type 3 IMS what information about the user is shared with another subject.

Figure 2.4 shows the three types of IMSes. Type 1 IMS learns the partial identity of a user accessing or using a system and derives information about physical identity. Type 2 IMS gathers attributes about each person, processes the information (during this phase, the information is typically aggregated) and outputs the results of a data analysis, such as graphs or trends in customer behaviour. Type 3 IMS is used by an individual to control the information that leaks to other parties, in other words, the individual manages [185] his or hers partial identities.

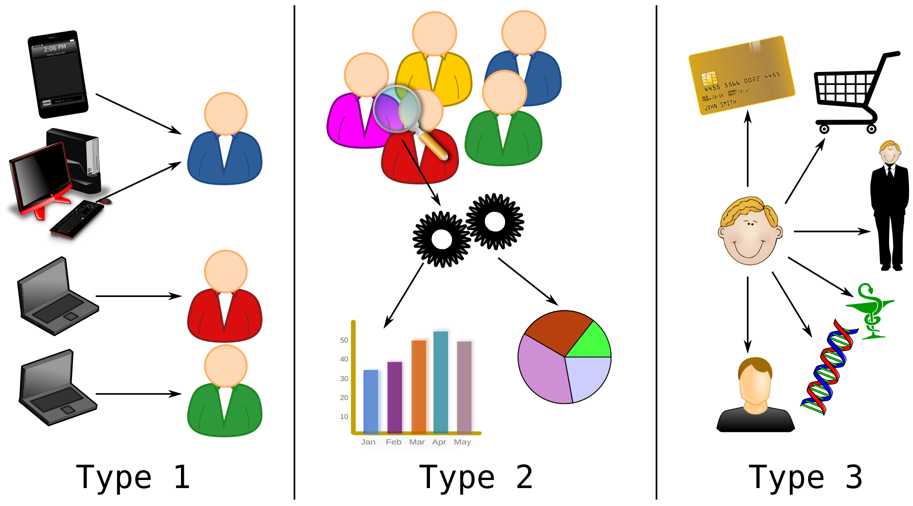


Figure 2.4: Three types of IMSes identified by FIDIS project.

Classification of identity management systems according to the role of identity management The other classification of IMSes developed by FIDIS project evaluates the role of identity management inside the system (typically a computer program). FIDIS distinguishes the three following *classes* [123, page 132]:

1. Identity management is the core functionality of the system (for example, an LDAP directory).

2. Although the identity management is an important part of the system, there is also other functionality that does not concern identity management (for example, a mail client that can handle more than one account).

3. The core functionality of the system is not focused on identity management, however, identity management has to be included for special tasks (for example, a web browser that can manage credentials to various websites).

2.3 Identification studied by this thesis

In this thesis, the goal is to identify human beings in computer networks. However, this goal cannot be achieved in all cases. Therefore, some methods discussed in this thesis aim at the identification of unique devices, leaving the identification of individual humans for further examination by forensic experts. Unfortunately, even the identification of unique devices cannot always be achieved. In such cases, this thesis lists possible shortcomings of the discussed methods.

The goal of the identification methods covered in this thesis is to infer the identity of the senders, receivers of messages transferred through the network.

In the context described by Nabeth [132], this thesis studies the identity mainly from the process perspective. For privacy reasons, we were not interested in gathering extensive amount of attributes and their values during this PhD research. The main goal of this research is to identify persons. However, to do so, it is necessary to be aware of partial identities and to study methods that link partial identities together.

SIMS [151] developed during this PhD research is a type 1 IMS. SIMS learns partial identities from computer networks (for example, by the partial identity detectors described in Chapters 6 and 7) and link them together by operations in identity graphs (see Chapter 8 for more details) to learn real-world identities.

Although SIMS is a type 1 IMS, it differs from a typical type 1 IMS as it does not focus on account management, authorisation, and accounting. SIMS focuses on a passive identification that is transparent to the authenticated user only. The partial identity linking employs methods that are also applicable to type 2 IMSes, see Section 9.4. Compared to a type 2 IMS, the developed IMS does not focus on gathering *huge amount of data* (in contrast to the perception of IMSes defined by Clauß and Köhntopp [34]).

SIMS is a class 1 IMS as its core functionality is identification. However, the applications concern class 2 IMSes as the aim is on the utilisation of the detected partial identities. The primary use case (see Section 9.1) deals with LI, during which all communication or metadata of the communication of suspects have to be collected. The IAN use case (see Section 9.3) utilises SIMS as service that links users to IP addresses (of their computers) and transport layer flows (produced by their computers). See Chapter 9 for additional details about applications of the proposed IMS.

Chapter 3

Basics of Networking

Computer networks are organised in layers. Each layer has its purpose and requirements to identify the communicating parties for successful message delivery. As previous studies observed [56, 65], current network protocols carry plenty of identity-related information.

This chapter briefly summarises TCP/IP stack and its layering. As this thesis focuses on methods applicable in IPv6, the necessary background required in following chapters is explained in detail. The main purpose of this chapter is to introduce the terminology and identifiers in network protocols. The last section explains specifics of the local and remote identification.

3.1 Network layers

Network protocols can be described with layered models, for example, ISO/OSI [99] or TCP/IP [20, 21]. Lower layers of the models deal with physical media and LANs. Middle layers provide means to route a packet across networks. Top layers deal with applications, provide rules for data encoding, session handling and other application-specific requirements.

Figure 3.1 depicts encapsulation in the TCP/IP model:

- At the start of communication, an application (such as instant messaging, web server, telephone) generates new data to be delivered to the other communicating party. The application usually supports at least one standardised or proprietary protocol suitable for the specific requirements of the application. The *application layer* protocol might prepend additional headers to the data to be exchanged between the communicating parties, for example, the time of the message or its encoding. Additionally, depending on the specific protocol, identifiers can be present, such as a username, *unique resource identifier* (URI), or chat room names. The resulting message is passed to the transport layer.

- The *transport layer* provides services such as application multiplexing, in-order delivery or congestion avoidance. These require additional data that are prepended in the form of a transport layer header. The multiplexing is performed using port numbers (carried in the transport layer header).

- Typically, there is not a direct connection between the communicating parties as each resides in a different location. The *network layer* enables communication across LAN boundaries. Typically, a single packet is routed through several intermediary devices, called routers, on its path from the source to the destination. A network layer header carries addresses that identify the communicating parties.

- LANs interconnect nearby hosts. Each LAN employs a specific protocol tailored for its medium (such as air, copper, fibre, their combination), the required bandwidth, and other requirements. These protocols are referred as *link layer* protocols. For correct packet delivery, each host is identified by a protocol-specific address carried in link layer headers, for example, MAC address.

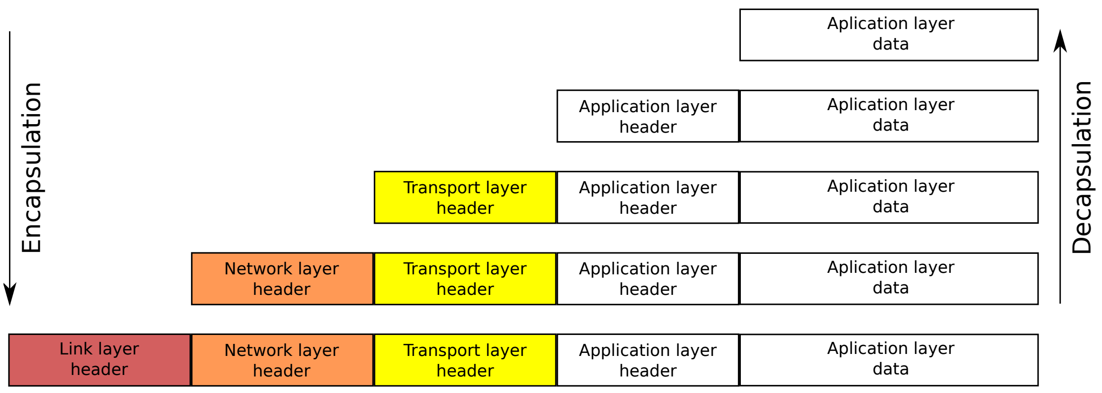


Figure 3.1: Encapsulation and decapsulation of data in the TCP/IP model.

The layering is often not as straightforward as depicted in Figure 3.1. The limited number of IP addresses in IPv4 is often solved by NAT that modifies network and transport layer headers, and sometimes even application headers or data.

The European FIDIS project studied [65, 123] network layers, network protocols and network identities. They raised concerns [65] that privacy experts do not participate in network protocols design. As a result, network protocols often rely on specific identifiers. These identifiers usually identify network equipment, applications or a person.

3.2 Identifiers in network protocols

The European FIDIS project provides an extensive list of network protocols covering each layer [65, Chapter 2]. As FIDIS noted, network protocols are often designed to maintain identifiers that identify the communicating parties. End hosts have addresses on both link (MAC addresses — 48 bits) and network layer (IP addresses — 32 bits for IPv4 and 128 bits for IPv6). Applications are addressed by ports on the transport layer. Individuals authenticate with usernames to access network accounts from different locations and devices.

Specifically, the FIDIS report [65, Chapter 2] focuses on:

- HTTP, FTP, SMTP, POP, and DNS (on application layer),
- TCP, UDP, SCTP (on transport layer),
- IPv4, IPv6, and IPsec (on network layer),
- Ethernet, PPP, IEEE 802.11 Wi-Fi, ISDN, Bluetooth, and DOCSIS (on link layer).

The FIDIS report [65] shows that virtually every commonly used protocol reveals information that is suitable to determine partial identities and link them together. Besides apparent identifiers, such as IP addresses, FIDIS report highlights the possibility of hidden identifiers that are present in network communication but that are not obvious.

Hidden identifiers appear as [65, Section 2.1]:

Protocol obscurities appear because protocol specifications often focus on interoperability. Hence, protocol specifications do not specify every detail to let the implementer pick the best solution for a specific platform. These details can leak identification information that identifies the platform. Section 6.2 studies protocol obscurities in ND implementations.

Misuse of protocol features occurs when a protocol parameter or a protocol option can be chosen from a domain freely. A server can use such parameters or options to identify a user session if all messages of the session carry the same or related values. Alternatively, an observer can use such parameter or option to link sessions executed over time, in case that messages of different sessions are marked by the same value (or a set of related values).

Manufacturing deviations are a result of physical constraints of a unique manufacturing process. It is not possible to build identical hardware on the atomic level. Consequently, these inconsistencies can be used to link data produced by a specific device. Chapter 7 studies properties of identification based on clock skew deviations.

The report [65] manifests that metadata, such as protocol headers, associated with current protocols leak many identifiers that can be directly or indirectly linked to specific persons. Moreover, cross-layer linkability reveals additional information about identities of specific devices or persons. However, even though the report mentions cross-layer linkability, FIDIS did not study the cross-layer linkability in detail. One of the goals of this thesis is to link identifiers found in network protocols and at the same time link the partial identities identified by the identifiers. Chapter 8 describes identity graphs that allow cross-layer linkability based on identifiers that appear in all layers of the TCP/IP model.

The report also notes [65, Chapter 1] that there is a difference between the specification of a protocol, which is usually in the form of a technical text document, and derived implementations of the specification. Sometimes an implementation diverges due to a misunderstanding of the specification. Sometimes a target platform contains constraints that prevent full conformance with the specification, for example, lack of resources in an embedded system. Often, some features of a protocol are optional and implementations may or may not implement such feature. Additionally, specifications of protocols often evolve. An implementation may follow an older version of the specification that is not in conformance with the latest version. Occasionally, there are conflicting requirements in a specification of a protocol.

Even though the report [65] does list many protocols on all layers of the TCP/IP model, it also admits that it is not a comprehensive compendium. This PhD research has focused on gathering additional information especially in the domain of IPv6, networks with NAT, and cross-layer linkability.

3.3 Internet Protocol version 6

As the FIDIS report [65, Chapter 2] notes, IP addresses identify either end hosts, or, in the case of NAT, the translator. This thesis focuses on identification methods related to IPv6. Sections 5.2 and 5.3 describe challenges in IPv6 and dual stack networks. Chapter 6 proposes the IPv6 addresses assignment detection. This section provides the necessary background.

IPv6 is a protocol that is constantly evolving, for example, see RFC 7721 [40]. Additionally, FIDIS project did not describe IPv6 address assignment in sufficient details. The following text expands the information about the IPv6 addressing and identification contained in the FIDIS report [65] with up to date information.

3.3.1 IPv6 addressing

IPv4 addresses are usually leased by DHCP. A device leases an IPv4 address from a DHCP server operated by the network owner. The device identifies its leasing interface by the MAC address carried in the *DHCP Discover* and *DHCP Request* messages. The device applies the leased IPv4 address to all its communication until the lease expires.

In contrast, IPv6 introduces several new mechanisms for address assignments. For example, *Stateless Address Autoconfiguration* (SLAAC) [184] is mandatory for all IPv6-enabled nodes in the network. SLAAC is a part of IPv6 *Neighbor Discovery* (ND). SLAAC allows an end device to generate as many IPv6 addresses as it needs, for example, for privacy concerns [11, 77, 135], as long as another device on the LAN does not use any of the addresses. The addresses are not leased but generated by end devices without any registration of the generated IPv6 address. The device concatenates the network part of the address, learnt from a router, with an auto-generated, usually 64-bits-long, *interface identifier* (IID) [42]. See Figure 3.2 for an example.

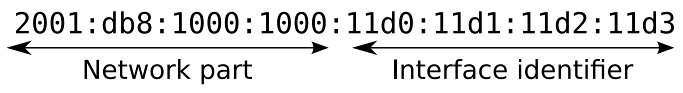


Figure 3.2: IPv6 address concatenates the network part of the address with an interface identifier.

Besides SLAAC, *Dynamic Host Configuration Protocol version 6* (DHCPv6) exists in IPv6 networks. However, the support for DHCPv6 is optional. In addition, even when active, the presence of the DHCPv6 server has to be signalled through ND messages and the leased IPv6 addresses have to be validated by ND.

In DHCPv6-controlled IPv6 networks, a *DHCPv6 Unique Identifier* (DUID) identifies a specific machine. Whenever a computer leases an IPv6 address from a DHCPv6 server, the computer identifies itself with the DUID. As a consequence, the DHCPv6 server in the network can link leases supplied to different interfaces of the same computer, for example, wired and wireless. However, whenever the operating system of a computer is reinstalled, or the machine is rebooted to a different OS, the DUID changes [82].

Cooper et al. [40, Section 4] provide a list of currently implemented mechanisms to generate IIDs:

IEEE-identifier-based IIDs are IIDs with an embedded link layer identifier, for example, the MAC address encoded as EUI-64 [42, 97].

Static, manually configured IIDs are IIDs selected and configured by the user of a computer.

Constant, semantically opaque IIDs are random but constant IIDs used by Windows instead of the IEEE-identifier-based IIDs. They do not leak MAC address to the Internet. Nevertheless, they do not change over time or whenever the device reconnects to a different network.

Cryptographically generated IIDs [10] incorporate a hash of the public key of a host.

Stable, semantically opaque IIDs [74] are random IIDs that remains constant in a specific network. They do change when the device reconnects to a different network.

Temporary IIDs [135] are random IIDs with limited life span.

DHCPv6 generation of IIDs are generated by a DHCPv6 server and can be temporary or non-temporary depending on the request created by a DHCPv6 client and the DHCPv6 server configuration.

Transition and co-existence technologies generate IPv6 addresses that usually embed an IPv4 address.

Once an address is assigned, it identifies the network interface of the machine that generated the address (or that obtained the address from DHCPv6).

3.3.2 Neighbor Discovery

For the understanding of the contribution of this thesis in the IPv6 identification realm, it is necessary to get familiar with the ND process.

ND [136] allows a node to discover routers and other hosts in the network. In addition, the node also learns prefixes in use in the network and other information. All ND messages are specific packet types of *Internet Control Message Protocol for the Internet Protocol Version 6* (ICMPv6).

ND depends on the multicast groups created on the LAN. There are role-specific groups, for example, all nodes (*ff02::1*), all routers (*ff02::2*). ND also depends on *solicited-node multicast groups* that are created automatically whenever a node generates a new IID. A solicited-node multicast group address is computed for an IID by concatenating the prefix *ff02::1:ff00:0/104* [89] and the lowest 24 bits of the IID. For example, the solicited-node multicast address for the address depicted in Figure 3.2 is *ff02::1:ffd2:11d3*. Switches on the LAN can treat multicast messages as broadcast or build multicast trees in case that they are able to perform *Multicast Listener Discovery* (MLD) snooping [30]. However, most current switches treat solicited-node multicast addresses as broadcast [141].

When a host connects to a network, it tries to discover routers in the network by issuing *Router Solicitation* (RS) requests. Routers in the network reply by a *Router Advertisment* (RA) [136, 196] message; either to the all host multicast group or by a unicast reply. RA includes one or more on-link prefixes and zero or more SLAAC prefixes configured in the network. Optionally, the RA instructs the host to get additional configuration from a

DHCPv6 server. Typically, DHCPv6 is not active. The host constructs IPv6 addresses from the advertised SLAAC prefixes and generated IIDs [184].

Before a newly generated tentative address can be used for communication, it is necessary to test its uniqueness in the network. *Duplicate Address Detection* (DAD) [127, 184] is a process during which the host sends a *Neighbor Solicitation* (NS) message to the special solicited-node multicast group [89] corresponding to the tentative address. In this thesis, NS issued during DAD is denoted as NS-DAD. If the tentative address is already configured on another host, the another host replies with a *Neighbor Advertisement* (NA) to the multicast group for all nodes in the network (*ff02::1*). If the tentative address is not already used by another device, there is no reply to the NS-DAD. The non-conflicting address changes state to either the preferred state or the deprecated state. The host can use the address for communication. To avoid race conditions in address assignments, RFC 4862 [184] orders that each host has to join the solicited-node multicast group before it sends the NS-DAD.

In Figure 3.3, DAD detects that another computer is using the newly generated address A. (1) Computer C1 subscribes to the solicited-node multicast group (corresponding to the IID of the address A [89]) by issuing an MLD Report. MLD capable switches and routers process the message. (2) Computer C1 sends NS-DAD for the address A. The NS-DAD is delivered to all nodes subscribed to the solicited-node multicast group corresponding to the address A (MLD snooping is active as depicted in Figure 3.3) or to all nodes (MLD snooping is not active). (3) As Computer C2 is already using the address A, C2 replies with NA to all hosts multicast group. Finally, C1 drops the address A.

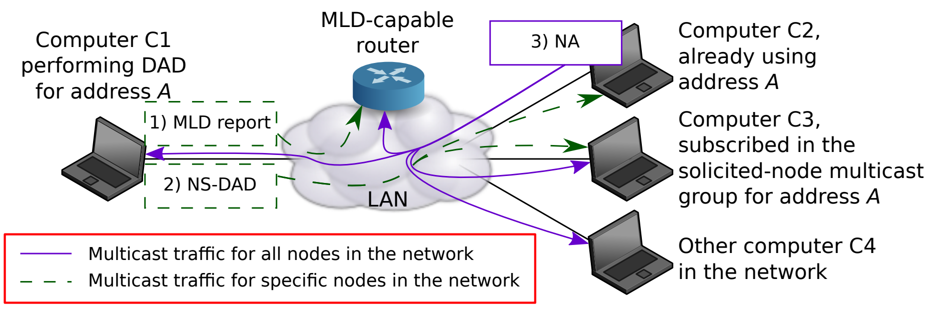


Figure 3.3: An example of DAD during which the computer C1 learns that the address A is already used in the network.

Optimistic DAD [127] allows usage of addresses with unique IIDs (such as a random IID, or an IID derived from a unique interface identifier) even before the DAD completes. An address in the optimistic state should not be used for communication, but if the host does not have any other option, it can use the address for communication. Note that the host performing the optimistic DAD cannot advertise the optimistically assigned address to other hosts before DAD completes and the address reaches the preferred or the deprecated state.

MLD capable routers periodically query all multicast groups for active subscribers. If any subscriber exists, one of them replies. In case that all IIDs containing the lowest 3 bytes of the solicited-node multicast group have been dropped, there is no MLD reply to the query. However, in the case of a collision in solicited-node multicast addresses, the group stays active while at least one of the colliding IIDs is active.

3.4 Local and remote identification

As the FIDIS report [65] highlights, some identifiers (for example, MAC addresses of end devices) are available only on the same LAN in which a device or a person to be identified are located. This thesis distinguishes between methods for local and remote identification. Local identification benefits from all identifiers, however, it is only applicable on the LAN of the device or the user to be identified. Remote identification restricts the information available to the network, the transport, and the application layer. Only in specific cases does a remote observer detect link layer identifiers, for example, in IEEE-identifier-based IIDs of IPv6 addresses.

Despite the name, the methods for remote identification are also applicable on LANs. However, their results can be less accurate compared to methods which focus on LANs.

In this work, we distinguish identities of specific subjects. Chapter 8 presents identity graphs that distinguish following sets of identifiers:

Link layer addresses of end devices identify a specific computer network interface. Some devices are operated dominantly by a specific person, in such case, it is possible to link a specific person with the device being used. Sometimes, it is possible to identify the person operating the device that has a specific link layer address by other means. For example, there are protocols, such as RADIUS [165], that authenticates the link layer of a particular device to a specific user account. Link layer identifiers only rarely leak [50, 77, 135] through a border of a LAN.

Network layer identifiers also addresses network interfaces of end devices, typically by IP addresses. The IP addresses were originally designed to be valid end-to-end. However, due to limited IPv4 space, IPv4 addresses are often translated. IPv6 address translation is not recommended [3, 32, 178]. However, when NAT is active in IPv6, it is recommended [33, Chapter 9] to map one inside IPv6 address to one outside IPv6 address. Therefore, IPv6 address identifies a computer while IPv4 address might identify a single computer or a set of computers. Nevertheless, an IPv6-enabled computer typically uses more than one IPv6 address simultaneously [135].

Transport layer flows can be identified unambiguously at a given moment with a 5-tuple: source and destination IP address, source and destination port and transport layer protocol type. Due to the presence of IPv4 address translation, it might be necessary to distinguish a traffic of computers behind NAT using transport layer flows. Besides network address translation, transport layer identifiers do not change for a specific flow. However, the 5-tuple has only limited validity time frame. Port numbers can be reused for different flows.

Application layer identifiers identify a specific partial identity of a person. Usernames (logins) are a typical example of such identifier. Additionally, application layer identifiers contain a domain name and consequently may identify the entity that registered such domain and is responsible for servers running the service; for example, consider web URI and e-mail addresses.

Chapter 4

Lawful interception

As the main topic of this thesis is to detect identities for LI, it is necessary to get familiar with the concept, laws, technology, related work, and the stakeholders in the LI. This chapter provides the necessary background. LI has been transformed from a loosely interpreted concept to a clearly defined set of law and regulations [187]. The description of LI and the architectures of LI systems primarily considers LI in the European Union and the Czech Republic in particular. Therefore, this chapter starts with an overview of the standards for LI maintained by the *European Telecommunications Standards Institute* (ETSI). This chapter also compares ETSI standards to the standards for LI in other parts of the world.

Figure 4.1 depicts high-level architecture for LI. An LI system is operated by an Internet service provider (ISP). The ISP either provides Internet connectivity or a specific service such as e-mail or voice over IP. The LI system has to identify intercept targets, their data, and pass the intercepted data to the *Law Enforcement Agency* (LEA) that initiated the interception. Intercepted data are typically stored at the *Law Enforcement Monitoring Facility* (LEMF) operated by the LEA that initiated the interception. Data are stored for further analysis and the forensic investigation by LEA agents.

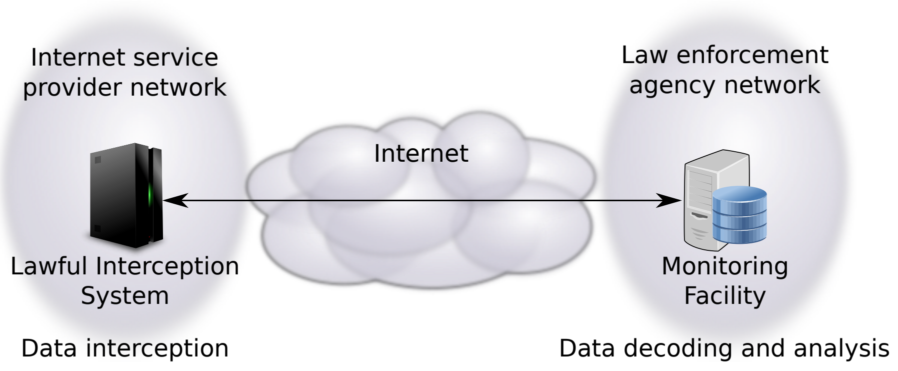

Internet service provider network
Law enforcement agency network
Internet
Lawful Interception System
Monitoring Facility
Data interception
Data decoding and analysis

Figure 4.1: Top level architecture of an LI system: intercepted data are passed to the LEA for further analysis.

Although the discussion about LI increased recently, for example, Telecommunications Industry Dialogue[1] was established in 2013, many details about the activities of government bodies remain secret [189], and the transparency varies country-by-country [188, 189].

[1] https://telecomindustrydialogue.org/

Typically, a small group of security-cleared personnel of an operator maintains an LI system and processes warrants [188]. Each warrant may target one or more subscribers while one subscriber may be covered by several warrants. The process varies according to national regulations.

The goal of this thesis is to propose mechanisms that identify the target of the interception according to the identifiers covered by a warrant. As Chapter 3 establishes, there are several layers, each of which uses different identifiers. Hypothesis 1 focuses on improvements in the identification on IPv6 LANs. Hypothesis 2 investigates linking of the IPv6 addresses outside LAN. Finally, Hypothesis 3 aims at the development of identity graphs — a formal model that links partial identities discovered in the network. Consequently, identity graphs enable targeted LI in conformance with the warrant and law.

4.1 Lawful interception standards and architectures

This section provides a survey of the principles of LI around the world. The basis of this section was published in Czech in a technical report of the Sec6Net project [159]. I am the original author of the text.

4.1.1 Lawful interception defined by ETSI

ETSI has standardised LI [53–63] for the European Union. ETSI specifies the LI framework and its basic parameters. Many specifics are left for national regulations[2]. The Czech Republic adopted the European legislation into its law order [118]. The warrant authorising an interception may require either interception of *intercept related information* (IRI) (metadata about the communication) or both IRI and *content of communication* (CC) (a copy of the communication of the target of the intercept — the suspect).

LI has to be transparent to the suspect. The traffic cannot be modified in any way during an interception. The network behaviour observed by the suspect has to remain exactly the same compared to the behaviour without the interception.

Each warrant has to specify at least the identifiers of a partial identity of a suspect, the duration of interception (the start time and end time), type of the interception (IRI or CC), and parameters for the delivery of the intercepted records.

IRI and CC records

There are four types of IRI records defined by ETSI [59]. Each type describes a specific set of activities performed by the target of an intercept. IRIs describe the communication of a suspect. An LI system generates an IRI record and passes the IRI record to the LEA.

1. A *begin* IRI record is created whenever a target connects to the network, generates a new IP address, initiates a new communication, or performs a similar action.

2. An *end* IRI record is created whenever a target disconnects from the network, stops using an IP address, finishes a communication, or performs a similar action.

3. *Continue* IRI records signal an acknowledgement of an open session, such as long-term usage of an IP address, or longstanding communication. A *continue* IRI record

[2]For more information, see, for example, Hoffman and Terplan [90], Harfield and Harfield [87] or the Law Enforcement Disclosure Report by Vodafone [189]

can also be used to give more details about a previously signalled communication, for example, when a lease of an IP address is prolonged.

4. Other messages, such as signalling of error conditions or attempts to connect to the network, are signalled by a *report* IRI record. IRI *report* is also sent when a target tries to access the network but the LI system cannot create IRI *begin* record for some reason.

An LI system generates IRI records based on the observed traffic that the intercept target transmits or receives as depicted in Figure 4.2. Before a session is initiated, IRI *report* is used to signal any information, such as session establishment parameters. Immediately after a session is opened, the LI system signals to LEA session parameters in IRI *begin*. Any additional information about the running session is sent as IRI *continue*. The LI system signals the end of the session by IRI *end*. Should any additional information arise, the LI system generates IRI *report*. Note that each IRI record (*begin*, *continue*, *end*, *report*) contains details about the event that has triggered the creation of the IRI record. In summary, the following regular expression denotes IRI records that are generated for a single communication session: *report* begin continue* end report**.

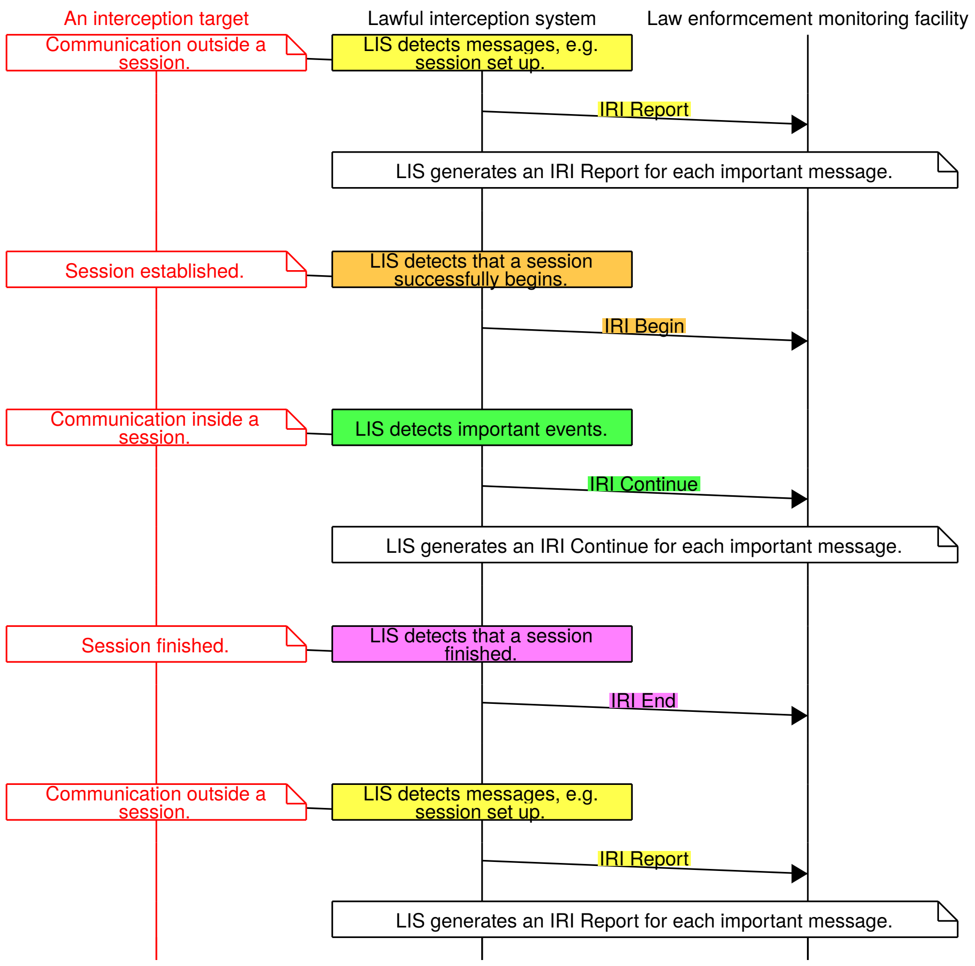

An interception target
Lawful interception system
Law enformcement monitoring facility
Communication outside a session.
LIS detects messages, e.g. session set up.
IRI Report
LIS generates an IRI Report for each important message.
Session established.
LIS detects that a session successfully begins.
IRI Begin
Communication inside a session.
LIS detects important events.
IRI Continue
LIS generates an IRI Continue for each important message.
Session finished.
LIS detects that a session finished.
IRI End
Communication outside a session.
LIS detects messages, e.g. session set up.
IRI Report
LIS generates an IRI Report for each important message.

Figure 4.2: IRI records provide metadata about communication sessions.

CC records, if allowed for an interception, are used to copy the communication of the intercept target. In this case, an LI system copies all packets sent by the target or destined to the target, including all protocol headers and application data. CC records are passed to the LEA.

Identification of the intercept target

Identification of an intercept target has to be unambiguous [58]. ETSI does not provide a fixed list of identifiers to be used for identification. However, ETSI lists [58] several identifiers that can be supported:

- Username or *Network Access Identifier* (NAI) [47]. NAI allows authentication during the access phase to the network.
- IP address (IPv4, IPv6).
- MAC address.
- Identifier of the access line or cable modem identifier.
- Other identifiers negotiated between ISP and LEA.

An LI system should generate IRI records in time to announce a change in the session state to the LEA. CC messages should be intercepted without any loss. Hence, an LI system has to identify the suspect as quickly as possible.

An interception warrant lists either (1) directly network-related identifiers that are the object of the interception or (2) other unique identification of the suspect by another identifier, for example, by his or her name and home address. In the latter case, authorised personnel of the ISP has to determine unique network-related identifiers of the suspect [59]. Typically, the authorised person learns the unique identifiers from an internal database of customers.

Reference model of LI system

ETSI created the LI system reference model [56] depicted in Figure 4.3. Communication between the LI system, the authorised personnel of the ISP, and LEA is carried over *Handover Interface* (HI) [53, 55, 57, 59, 60]. HI is divided into three parts — HI1, HI2, and HI3:

- LEA inserts and removes intercepts through the HI1 interface. Additionally, through HI1, ISP informs LEA about initiation and termination of the interception requested by the LEA or any technical issues concerning the interception requested by the LEA.
- The HI2 interface carries IRI records from the ISP to the LEMF of the LEA that initiated the intercept.
- The HI3 interface carries CC records from the ISP to the LEMF of the LEA that initiated the intercept.

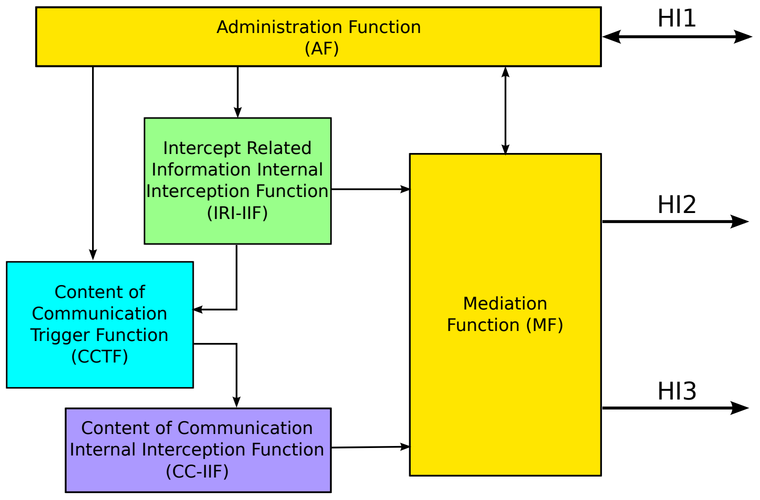

Administration Function (AF)
HI1
Intercept Related Information Internal Interception Function (IRI-IIF)
Mediation Function (MF)
HI2
Content of Communication Trigger Function (CCTF)
HI3
Content of Communication Internal Interception Function (CC-IIF)

Figure 4.3: ETSI reference model for LI [56]

The ETSI reference model accompanies five cooperating functions:

- *Administration Function* (AF) manages intercepts in progress. It configures other blocks whenever an interception starts or finishes.
- *Intercept Related Information – Internal Interception Function* (IRI-IIF) learns the identifiers that appear in the network. Additionally, IRI-IIF creates IRI records related to the intercepts in progress.
- *Content of Communication – Internal Interception Function* (CC-IIF) intercepts CC records. CC-IIF creates a copy of the packets transferred through the network.
- *Content of Communication Trigger Function* (CCTF) controls CC-IIF and selects *Intercept Access Point* (IAP); IAP is the point in the network where data are intercepted.
- *Mediation Function* (MF) passes IRI and CC records to LEMF. Additionally, MF can correlate IRI and CC records.

Figure 4.4 shows an example of the communication inside the reference model of an LI system. LEA delivers requests for interception through HI1, either manually or electronically [59]. Authorised personnel insert the intercept to the AF. ETSI requires decoupling of HI1 and internal interfaces so that LEA cannot control an LI system remotely [59]. The authorised personnel is only allowed to insert intercepts authorised by a court.

AF inserts a new intercept to a queue of intercepts to be activated. The intercept is activated after its start time passes. Then, AF configures other parts of the system for interception and later deactivates the interception.

Identifiers related to the partial identity of the suspect can change over time. IRI-IIF detects such changes either from network traffic or through other means, for example, by log analysis. Each change of a partial identity of a suspect is immediately signalled to MF by IRI records. IRI-IIF also dynamically configures CCTF with dynamically detected identifiers.

CCTF receives static and dynamic configuration for intercepts with CC interception enabled. AF handles the static configuration while IRI-IIF provides dynamic configuration. CCTF controls CC-IIF probes and determines the configuration of each probe separately.

CC-IIF usually consists of several CC-IIF probes, each of them is located on a particular link where it intercepts data according to the identifiers observed in packets. CC records are created from the copy of the incoming traffic.

MF can correlate IRI and CC records or change their encoding according to the desired format of the intercepting LEA. MF sends IRI records through HI2 and CC records through HI3.

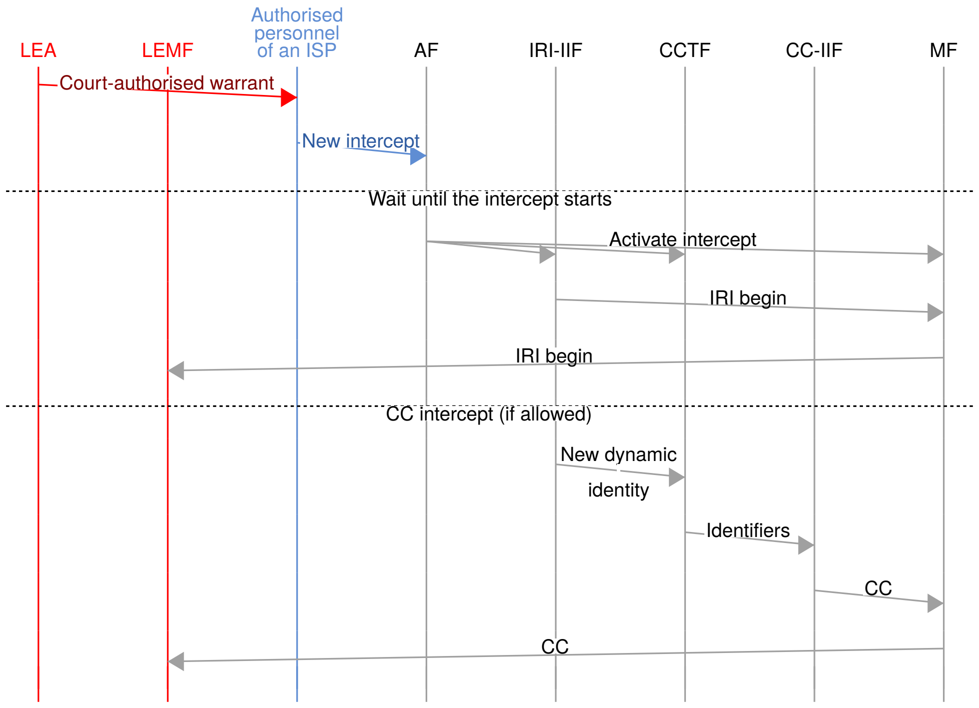


Figure 4.4: An example of communication within the ETSI reference LI system.

4.1.2 ATIS/TIA J-STD-025 standard

The J-STD-025 [9] standard specifies LI in the United States. J-STD-025 is coupled with the law called CALEA [37] that regulates LI in the United States.

J-STD-025 specifies the following three interfaces [90] to an LI system:

- *Surveillance Administration System* (SAS) provides ports to the system to be used by LEAs.
- *Call Data Channel* (CDC) provides communication metadata related to establishing, maintaining and terminating communication sessions.
- *Call Content Channel* (CCC) delivers a copy of the communication content.

Roughly, SAS is analogous to HI1, CDC is analogous to HI2, and CCC is analogous to HI3. A J-STD-025 LI system cooperates with the monitoring facility of the LEA.

Figure 4.5 displays the J-STD-025 reference architecture. The components are:

- *Lawful Authorisation* provides warrant authorization by a court in conformance with CALEA. The warrant is forwarded to the Telecommunication Service Provider after it is authorised.
- *Telecommunication Service Provider* (TSP) is a provider of telecommunication services of a suspect. An LI system is deployed in the network of the TSP. An LI system is composed of the following blocks:
  - *Service Provider Administration* manages intercepts and configures other parts of the system.
  - *Access* block connects the parts of the LI system that are not a part of the TSP's infrastructure to the production network of the TSP.
  - *Delivery* block encodes intercepted data to the format desired by LEA.
- *Law Enforcement Agency* monitors the intercepted data from the TSPs carrying interception. The monitoring facility is divided into two parts:
  - *Law Enforcement Administration* manages the intercepts and configures storage devices for the intercepted data.
  - *Collection* stores both metadata and the copy of the content of communication of suspects.

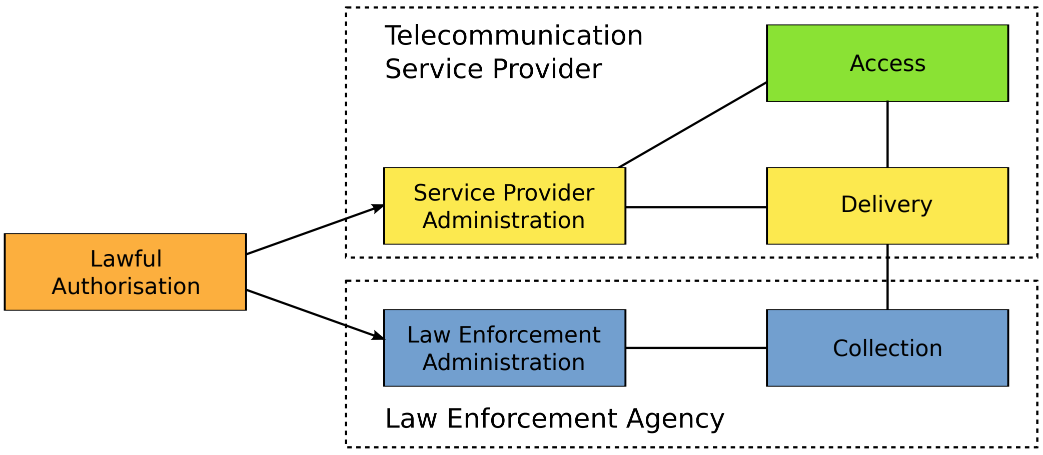

Telecommunication Service Provider
Access
Service Provider Administration
Delivery
Lawful Authorisation
Law Enforcement Administration
Collection
Law Enforcement Agency

Figure 4.5: J-STD-025 [9] standard LI architecture.

4.1.3 Cisco architecture for lawful interception systems

RFC 3924 [13] describes the architecture of an LI system providing a minimal set of features required by different national regulations of LI. The architecture was developed by Cisco Systems[3], and it reflects the requirements arising both from ETSI standards and J-STD-025. The handover interface between the LI system and LEA is very similar to the one

[3]http://www.cisco.com

defined by ETSI. Nevertheless, it is configurable to be compatible with J-STD-025. Cisco reference architecture is divided into the following parts (see also Figure 4.6):

- *LI Administration Function* manages intercepts and configures other parts of the system.
- *Intercept Access Point* (IAP) is a point in the network where an LI system connects to the regular infrastructure of an ISP. Alternatively, IAP may directly represent devices carrying interception. There are two types of IAPs:
  1. *Content IAP* captures traffic.
  2. *IRI IAP* creates IRI records (metadata about the traffic of a suspect).
- *Mediation Device* (MD) configures IAPs, replicates intercepted records in a case that one subject is monitored by multiple LEAs, transcodes intercepted data to the format required by the LEA, and relays the intercepted data through HI2 and HI3 interfaces.

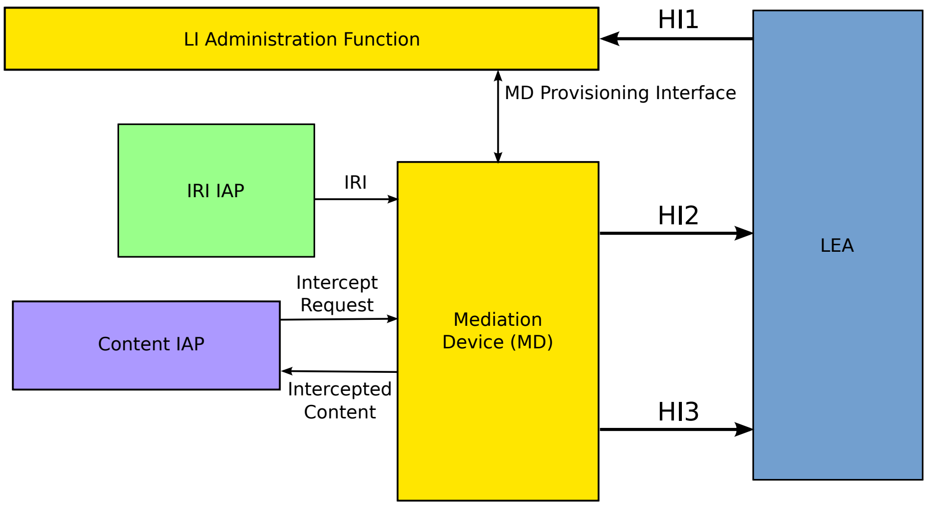


Figure 4.6: The LI system architecture published in RFC 3924 [13].

4.1.4 Omnipresent surveillance

LI denotes domestic targeted interception of individuals or small groups. Many countries balance between the right to privacy and the need to investigate crimes [188]. However, some jurisdictions mandate that operators intercept all international traffic going through their network [188] or allow warrantless wiretapping of specific international communication [115].

Recently, several indices about mass-scale surveillance program in the United States of America called PRISM leaked to the public [119]. It is a surveillance program enabled [119, 138] by Foreign Intelligence Security Act of 1978 with amendments of 2008 and U.S Patriot Act. After the disclosure, the debate about the program started, and official groups were established[4] to limit [138, 183] the surveillance.

[4] http://icontherecord.tumblr.com/faq

Another example of massive scale interception comes from Georgia [177]. In 2013 a law mandating direct network access for LEAs emerged.

Such mass-scale surveillance can harm the Internet and other technology companies [64, 119]. Telephone and communication operators try to persuade [177, 188] governments to follow ETSI standards for LI. Google started to encrypt its traffic between data centres [71], and it promotes encrypted communication[5].

4.1.5 Relations of the LI architectures and this thesis

Recent events seem to favour targeted surveillance while increasing the need to identify the originators and receivers of Internet traffic. This PhD research focuses on the development of detection mechanisms compatible with ETSI standards. However, due to many similarities between the ETSI LI, J-STD-025, and RFC 3924 described in this section, the results of this PhD research are applicable to ETSI LI, J-STD-025, and RFC-3924-based LI.

The identification mechanisms described in this thesis are compatible with targeted surveillance. The aim is to enable LI in modern networks.

4.2 Lawful interception systems

Only a limited amount of information about LI systems is available for general public, for example, in white papers of LI systems vendors [7, 187]. The available white papers and technical description indicate that some open problems exist. This section summarises public knowledge about LI systems; Chapter 5 focuses on open challenges.

4.2.1 Research of lawful interception methods in academia

Hoffman and Terplan [90] list many complications specific to IP networks. Among them is the need to extract source and destination partial identities embedded in the data flow, numerous IP connections are interleaved, Internet links are often deployed in an ad hoc manner, distinguishing the target and the nontarget data is paramount. This PhD research focuses on the identification of packets in IP networks. Identity graphs are constructed from several partial identity detectors of information with the goal of separating partial identities of the target from the partial identities of benign users.

In academia, one branch of LI research [108, 125, 195] studies *voice over ip* (VoIP) protocols. Karpagavinayagam et al. [108] proposed alternative LI architecture. They redirect both signalling messages and data messages of VoIP through an LI system. The LI system modifies signalling messages so that the voice channel takes a predictable path. However, the modifications do not comply with ETSI requirements on LI that prohibit any modifications of traffic to be intercepted as this could reveal the intercept in progress. Milanović et al. describe methods [124] and distributed deployment [125] of a VoIP LI system. Their solution is compliant with LI standards. Compared to this thesis, their solution to identify users is also distributed, but they focused only on one protocol, specifically H.323. The architecture described in this thesis is generic, and it is suitable for identifiers from all networking layers.

Another branch in academia is focusing on the reliability of LI. Bhargavan et al. [17] studied protocol obscurities [65, Section 2.1] the behaviour of mail servers in the presence of unusual requests, for example, requests that are not completely specified in protocol

[5] https://www.google.com/transparencyreport/

specifications. They found that the processing of the requests depends on the server-side software and its version. Consequently, an attacker can send unusual messages to trick an LI system.

Cronin et al. [44] studied LI and identified several methods that a tracked person can employ to evade or confuse the identification of their data. Confusion and evasion are defined as follows:

Confusion is a process during which an intercept target tricks the intercepting device or a data analyser into decoding a different message than the one that was delivered to the recipient. Although during confusion the investigator detects the communicating parties, the false message can mislead the investigation. For example, an attacker can transmit two TCP segments with the same sequence number but different content [142]. As Cronin et al. shown, current software interprets such messages in different ways [44].

Evasion is a process during which an investigation target completely hides the fact that any communication occurs; the intercepting device does not detect any communication. For example, on the physical layer, the interoperability of several devices is achieved by specifying some tolerance, for example, in frequency, voltages. An LI target can construct a device that sends messages just outside the tolerance range. Due to the nature of analogue signal processing, some devices can process such signal but others cannot [44]. If the monitoring system does not detect such transmissions but the next hop does, an LI target evades the monitoring.

Research in the field of network intrusion systems described traffic normalisation [83] that removes the ambiguity that causes confusion. However, traffic normalisation modifies the traffic, and consequently, it can be detected by an LI target. Hence, traffic normalisation cannot be applied in LI. In this thesis, the confusion is minimised by expecting the deployment of partial identity detectors as close as possible to the accessed service providing the partial identity. On the other hand, CC-probes should be located as close as possible to the intercept target [157].

Bellovin et al. [15] list and evaluate the effects of alternative methods for the electronic gathering of evidence: legalised hacking of devices belonging to suspects through existing vulnerabilities in end-user software and platforms. In this scenario, law enforcement does not get evidence from the network but instead deploys monitoring software directly to the computer of the suspect. Consequently, law enforcement needs an invisible, yet reliable way to penetrate a device of a suspect. This work does not consider such methods as it is not allowed by ETSI standards for LI.

Rojas et al. [166] focus on LI and mobility in IPv6 networks. They present a mechanism that predicts the movement of an intercept target. The goal is to minimise the number of probes performing the interception. The work of Rojas et al. is orthogonal to our work. This PhD research focuses on identification mechanisms. Once the intercept target is identified, the approach of Rojas et al. can be employed.

4.2.2 Proprietary lawful interception systems

Major network equipment manufacturers provide equipment with LI capabilities. For example, Cisco supports LI functions as an optional feature. Cisco follows the architecture [13] described in Subsection 4.1.3. However, the communication protocols are proprietary, and they are not available for general public.

Aqsacom offers *Aqsacom real time Lawful Interception System* (ALIS)[6]. Aqsacom published a white paper [7] that describes LI and ALIS. The white paper also describes challenges for LI, such as the problem of distinguishing data of the intercept target and data of other people that are not subject to the intercept, for example, neighbours sharing an Internet link. Another problem is ambiguous law in several countries. The downside of the white paper is that it does not provide the complete overview, for example, it completely omits SLAAC [135] in IPv6.

Hoffman and Terplan [90, chapter 6] describe ALIS. They note that ALIS supports many network-related identifiers including IP address, MAC address, and user IDs, such as SIP identifiers and e-mail addresses.

Utimaco[7] is another company that published a white paper on LI [187]. It lists several challenges for LI: (1) the diversity of access technologies, such as ISDN, xDSL, WLAN, WiMAX, GSM, GPRS, UMTS, CDMA, and cable; (2) voice communication that progressed from fixed PSTN to mobile networks, VoIP, and other specialized applications; (3) high speed networks and exchange points carrying tens of gigabits of data per second. This thesis focuses on LI in IP networks. Identity graphs defined in Chapter 8 support various partial identity detectors and identifiers that appear in all layers of the TCP/IP model. During this PhD research, more than 15 partial identity detectors were created [22, 70, 91, 92, 103, 159, 173]. High-speed networks are not a core part of this thesis. Nevertheless, the Sec6Net project also focused on high speed networks up to 10 Gbps, and SLIS [152, 159] supports CC-probes for 10 Gbps networks (see also Chapter 9).

SS8[8] designed LI system called Xcipio modularly [90, chapter 6] so that it is prepared for constant changes in the networking environment. Xcipio supports many protocols on different network layers including VoIP. Identity graphs defined in Chapter 8 are also modular and support various partial identity detectors.

Other vendors of LI systems also exist: for example, Verint, IP Fabrics VSS monitoring. However, the exact capabilities of their solutions are typically not public.

4.3 Stakeholders

LI impacts many areas including privacy, security, crime investigation, and private companies. As a result, many stakeholders, which can be interested in the results of this thesis, exist. This section identifies the stakeholders and their relation to LI.

Even though this thesis focuses on technical aspects of LI, it has to provide the context so that a reader is aware of potential risks and controversy.

The description of the stakeholders in the area of LI and their interests follow:

Policy makers, governments and regulators have a conflicting role. On one side, they need to respect the privacy of individuals [138, 183], for example, fundamental rights and freedoms given by constitution and international treaties [39]. On the other side policy makers and governments have to prepare a law that protects the society from criminals and terrorists [115, 129]. Both goals have to be bal-

[6]http://www.aqsacom.com/en/lawful_interception.aspx
[7]https://lims.utimaco.com/products/lims-access-points/
[8]https://www.ss8.com/what-we-do/intel-law-enforcement-investigations/

anced [122, 187], and there should be safeguards that prevent to minimise the privacy infringement [113, 187]. The balance should be clear for general public [113].

Investigators are typically strictly bound by the law introduced by policy makers. They need to respect the established balance between the right to privacy and the right to safety. They have to weigh [87] the importance of getting information for ongoing investigations and privacy violations caused by data interception and decoding. However, investigators are monitored by courts and prosecutors. Hence, investigators have to describe the proportionality of surveillance to a court including the decision between interceptions of only metadata or also the copy of the message content [115]. Sometimes, the content is less important than the fact that two parties communicated with each other [115]. When a court approves an intercept, investigators need accurate and indisputable evidence [26], so that the collected evidence is admissible for trials.

Intelligence agencies form a special case of investigators. Often, intelligence agencies deal with international threats. Consequently, they often have to emphasise the right to safety [90]. Their complex situation was proven by the PRISM case [119]. Too-wide surveillance hampers the trust and can influence business [64]. Currently, there are efforts to limit mass-scale surveillance and increase transparency [122, 138].

Prosecutors in some countries, for example, the Czech Republic, authorise the scale of an LI. They oversight the actions carried by investigators and intelligence agencies and balance between the right to privacy and the right to safety.

Judges and courts are the most influential bodies in searching for the balance between the right to privacy and the right to safety as their decision is often ultimate. To decide trials, they need sound arguments and indisputable evidence [26].

Telecommunication companies and internet services providers have to obey local law and cooperate with investigators and intelligence agencies. At the same time, enterprises have to invest and make the good reputation for making a profit. Thus, they need to care about the privacy of their users as the profitability of their business is strongly influenced by public opinion. Recently, telecommunication industry companies formed alliances, such as Telecommunication Industry Dialogue, which tries to raise awareness about the goals of the industry. Some companies also release reports [76, 188] about law enforcement investigations.

Intelligence support systems vendors need to provide tools to enable investigation and yield reliable evidence [113]. They react to the needs of investigators and intelligence agencies. Often, the needs are very specific due to the potential violation of privacy rights. Hence, the cost of the *intelligence support systems* (ISS) is usually high. Typically, companies do not specialise in ISS. If these companies cooperate with intelligence agencies too much, they might face strong criticism from the public [119] or even lose profit [64].

All private sector companies are in the risk of technology stealing [115]. ISS manufactured to protect society are sometimes [115, chapter 8] misused for industry espionage. Companies need to define policies to protect their data [115, chapter 9], which increases their expenses and consequently lowers their profit.

During the *Athens Affair* [115], Vodafone Greece purchased telephone switches with LI capabilities inactive. However, unknown intruders managed to exploit the LI functionality to spy on politicians and other prominent persons. In the end, the reputation of Vodafone Greece went down due to the vulnerability in the switches introduced by their vendor, Ericsson. Hence, LI functionality (as any other functionality) in network equipment comes with higher risks for bugs and vulnerabilities that may even hurt products without the functionality if the design of the two product lines is not separated correctly. The manufacturers have to accept additional costs for maintaining LI functionality in their products.

The general public is concerned about privacy and security of private data of citizens [190]. However, the general public is also aware of security risks and possible crimes. The general public needs to feel safe and see that police can cope with crime. At the same time, the general public does not want their confidential data and communications to be stolen. These two opposing needs have to be balanced. Private data interception has to be regulated, and a strict law concerning LI has to be in place to achieve both goals.

Privacy experts often form an opposition to the roles of investigators and intelligence agencies. Their primary aim [38] is in the protection of privacy rights. They are trying to reach their goal by disseminating their views to the general public, policy makers and governments.

It is evident from the list of goals of the stakeholders that there are several challenges on different levels — technical, ethical, financial. This thesis does not deal with ethical issues, and it does not balance the rights to privacy and the right to safety. These issues are ethical, and their solutions cannot happen on the technical level. Instead, in the area of LI, this thesis focuses on tools providing information for investigators. In the area of privacy, this thesis aims at distinguishing partial identities of specific persons. Consequently, the scope of intercepts may be limited only to the traffic of suspects, leaving the traffic of benign citizens untouched.

This thesis studies new techniques for partial identity detection and lists their properties. Any user of the techniques described in this work should consult local legislation and determine if the techniques are applicable in the specific jurisdiction.

4.4 Research of lawful interception

This thesis focuses on LI in the context of the Czech and the ETSI standards. The study of requirements on LI revealed several parameters of identification methods to be applicable in LI:

- As the interception has to be transparent to the suspect, the identification methods cannot modify network traffic.
- The identification of the suspect has to be as quick as possible so that IRI *begin* can be created and CC intercepted.
- The identification method should detect both session start (IRI *begin*) and end (IRI *end*).

Additionally, CC interception is typically based on IP addresses as the IP address of the intercept target is present in all IP datagrams of the target [13]. However, the IP addresses are often assigned dynamically. The authorised personnel of an ISP can activate intercept based on identifiers such as RADIUS username or the identifier of the access line. Hence, an LI system has to be able to link the input identifier to the IP addresses assigned to the intercept target.

As a part of the Sec6Net project, I managed a group of students with a goal of developing a proof-of-concept LI system. The main goals of the group were:

1. To study existing methods for identification of network subjects and to propose new methods applicable to modern and future networks.

2. To create a framework that controls network probes developed as a part of the Sec6Net project.

As a result, we developed Sec6Net Lawful Interception System (SLIS) [152] that reached the goals mentioned above.

This thesis describes the steps needed to reach the first goal. Chapters 6 and 7 focus on specific methods for partial identity detection including related work. Chapter 8 focuses on partial identity linkage that helps in getting complete information about partial identities of a specific LI target. Finally, Section 9.1 presents the application of the methods developed as a part of this thesis in SLIS.

Chapter 5

Challenges of identity detection in modern computer networks

Computer networks connect various computers together. User behaviour and typical usage of network change with rising number of interconnected devices [86]. Communication infrastructures became distributed, highly complex and service-oriented [185]. Consequently, network protocols evolve. With new protocols, new challenges for identity detection emerge.

In the past, a typical household owned a single desktop computer. The advent of laptops, smartphones, tablets, wearables and other network-enabled small devices increased the number of devices operated in a single household. Nowadays, shared computers are less common. Often, a person exclusively uses several devices that access the Internet and browse web pages.

As a result, the knowledge that a device produced some traffic is often sufficient to link the traffic to the person that exclusively uses the device. However, as the person typically owns several devices, the traffic of a single identified device represents only a fraction of the traffic produced by the owner. Hence, the identification of a particular device represents only a partial identity of its owner.

IP addresses are prominent device identifiers [13, 26]. However, one of the biggest problems in networking in the last years is the depletion of the IPv4 address space [168, 182]. Nevertheless, IPv4 is still dominant protocol used on the Internet [109]. The shortage of IP addresses is countered by NAT, which typically hides many devices behind a single IP address or a pool of IP addresses. The share of IPv6 traffic is increasing. As already reported [81], the identity detection in IPv6 is different compared to IPv4.

This chapter lists challenges of user identification in current networks with a focus on LI; each section represents one challenge. This chapter (and thesis) considers both IPv4 and IPv6. The final section of this chapter explains the relation of this thesis to the challenges listed in this chapter.

This chapter is based on the ideas presented in papers *On Identities in Modern Networks* [157] and *Challenges in Identification in Future Computer Networks* [146]. I am the author of the text in both papers.

5.1 Network address translation

The first challenge for identity detection in current networks concerns NAT. A typical household customer buys a connection to the Internet from a local ISP present in the area of the household. The ISP is typically local or nationwide. An ISP is connected to the Internet either at an *Internet Exchange Point* (IXP)[1] or through another transit ISP (typically small ISPs).

The ISP typically forms a contract with one representative of the household who becomes (de jure) the customer. However, for LI, it is important [7] to identify the traffic of a specific member of the household as the person who signed the contract does not have criminal liability for the content of the traffic produced by other members of the household.

Until recently, a typical household received one public IPv4 address. All members of the household shared the IPv4 address as the home network was connected to the ISP with a router providing NAT. NAT maintains a table that defines a bijection between local and global identifiers of the flows traversing the network boundary. Hence, all traffic of the household used to be identified by a single global IP address. However, the depletion of the IPv4 address space does not allow ISPs to assign a public IPv4 address to all customers (if the ISP does not have enough IPv4 addresses from the past). As a result, ISPs employ a second layer of NAT for the traffic destined to the Internet. The two-layer network address translation is called *Carrier Grade NAT* (CGN). Figure 5.1 shows an example of an ISP employing CGN.

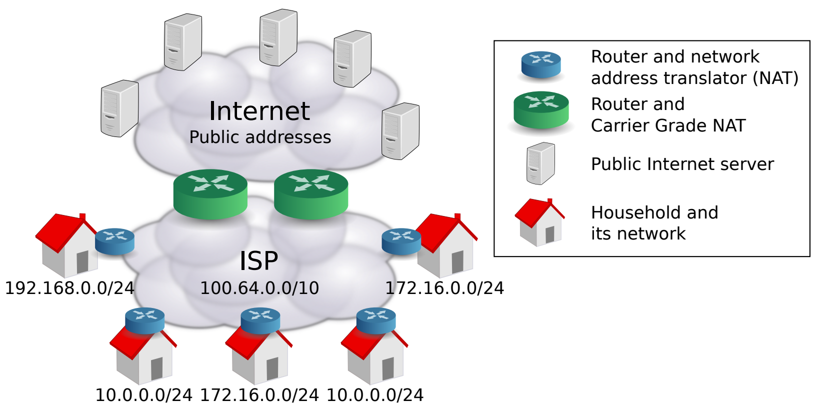


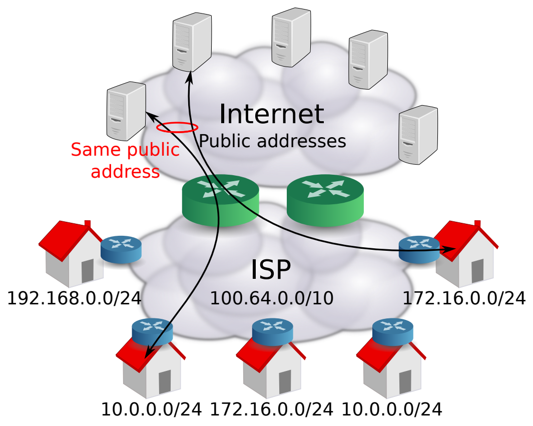
Figure 5.1: Nowadays, ISPs often employ CGN in their networks.

Each household in Figure 5.1 operates a LAN, on which local machines use the address space reserved by RFC 1918 [163] for LANs. The addresses are not globally unique. Consequently, the address space of many households collide. NAT at the boundary of the household network translates the local IP addresses to a single IP address provided by the ISP from the address space used in the ISP network. The ISP network should use addresses from the shared address space reserved by RFC 6598 [193]. CGN at the edge of the ISP network typically uses a pool of public IPv4 addresses. It is no longer possible to guarantee neither that all traffic of a single household can be identified by a single public IPv4 address nor that all traffic identified by a single IPv4 address belongs to the same household.

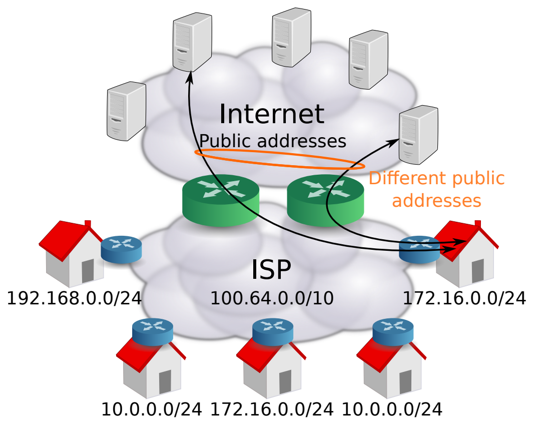
Figure 5.2 shows the challenges in identification of traffic origin arising in a CGN-enabled network. In the first example, users from two different households try to reach the external servers at the same time. It is possible that CGN translates both requests to a single public IPv4 address. The second example shows that the traffic of a single household can be identified with different public IPv4 addresses. In this case, the traffic is routed through distinct translators that do not share the same public address space. Note that the exact

[1]For example, http://nix.cz

behaviour of CGN in both cases is not specified as it depends on the network configuration and state.



a) Traffic of two household can share the same public IPv4 address.



b) Traffic of a single household is not identified with a single IPv4 address.

Figure 5.2: Challenges in networks with CGN.

Some sources, for example, Casey et al. [26], treat an IP address as a unique identifier of a computer attached to the network. The examples of CGN mentioned above show that this is not the case anymore. NAT can hide several machines behind a single IP address, CGN can hide thousands or even millions of households behind a single pool of IPv4 addresses. Hence, an IPv4 address identifies neither a single machine nor a single household.

LI warrants unambiguously specify the traffic to be intercepted. Depending on the exact wording of a warrant, it can be legally forbidden to capture all traffic of a single IP address when the court gives permission to intercept traffic of a single person, and the person is connected behind NAT. Therefore, LI performed in a CGN network needs to take the multiple address translations into account.

This thesis tackles this challenge with NAT-aware rules for the construction of identity graphs proposed in Chapter 8. The idea is to observe the communication before and after NAT performed by a translator, either by probes located before and after the translator [80, Chapter 4] or from logs generated by the translator. Another option is to correlate traffic of a single machine behind NAT through another mechanism, such as the clock skew [112]. The resulting identity graph differentiates between global and local identifiers.

5.2 Addresses in IPv6 Networks

IPv6 penetration in the Czech Republic is one of the highest in the world [67]. Thus, it is necessary that an LI system deployed in the Czech Republic identify IPv6 users. This section focuses on IPv6 addresses and the IPv6 address space. Section 5.3 outlines challenges related to the coexistence of IPv4 and IPv6.

The IPv6 address space is much larger compared with the IPv4 address space. In contrast to IPv4 where a computer network interface is usually identified by a single IPv4 address, in IPv6, each interface typically uses several IPv6 addresses [35].

Another difference between IPv4 and IPv6 deployment is the presence of NAT. Whereas multiple layers of NAT do occur in IPv4 (see Section 5.1), IPv6 networks usually do not employ NAT as the NAT-free network allows end-to-end connectivity [178]. Moreover, “the IETF does not recommend the use of Network Address Translation technology for IPv6” [192].

Section 3.3 provides an introduction into IPv6 and different kinds of IIDs. For correct IPv6 functionality, for example, for ND, each interface has to configure a link-local IPv6 address from the `fe80::/64` address space. Typically, the IID of the link-local address of a specific interface is stable. Nevertheless, traffic from a link-local address is not routable [89, Subsection 2.5.6]. Therefore, the operating system of the computer typically generates additional addresses. SLAAC is the default mechanism to obtain IPv6 addresses, DHCPv6 is optional. Current operating systems (Windows, Mac OS X, iOS, user-friendly Linux distributions) generate both stable addresses and temporary addresses. Temporary addresses are random and periodically regenerated. Windows, Mac OS X, iOS, and Ubuntu generate a new temporary address daily. However, the Wi-Fi network reauthorization or a network cable disconnect and reconnect often triggers regeneration of temporary addresses. Consequently, a single computer can use many IPv6 addresses during a single day even if a user of a computer does not perform any intentional action aimed to regenerate temporary IPv6 addresses.

Figure 5.3 shows an example of a single computer that generates several IIDs for a single interface.

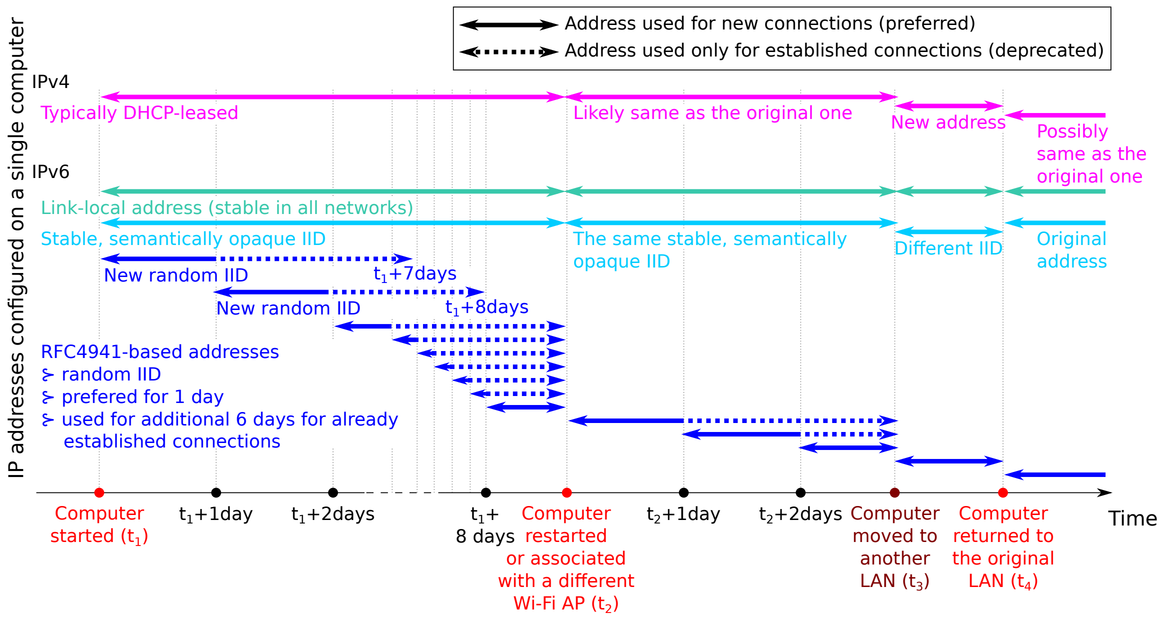

Address used for new connections (preferred)
Address used only for established connections (deprecated)
IP addresses configured on a single computer
IPv4
Typically DHCP-leased
Likely same as the original one
New address
Possibly same as the original one
IPv6
Link-local address (stable in all networks)
Stable, semantically opaque IID
The same stable, semantically opaque IID
Different IID
Original address
New random IID
t₁+7days
New random IID
t₁+8days
RFC4941-based addresses
⊱ random IID
⊱ prefered for 1 day
⊱ used for additional 6 days for already established connections
Computer started (t₁)
t₁+1day
t₁+2days
t₁+ 8 days
Computer restarted or associated with a different Wi-Fi AP (t₂)
t₂+1day
t₂+2days
Computer moved to another LAN (t₃)
Computer returned to the original LAN (t₄)
Time

Figure 5.3: A single network interface usually operates several IPv6 IIDs while it uses only one IPv4 address.

1. The computer in Figure 5.3 starts at time t_1.

   (a) The operating system automatically generates a link-local IPv6 address belonging to the `fe80::/64` network. As explained above in this section, each IPv6 enabled interface has to configure a link-local address for service communication, such as ND. Nevertheless, the address can be used for any local communication.

   (b) The default IPv6 address assignment method is SLAAC, a subset of ND. Let us consider that the computer learns the prefix used in the network for address autoconfiguration [184] from ND. For each prefix learnt from ND, most current operating systems (Windows, Mac OS X, iOS, most Linux distributions) configure:

  - a persistent IPv6 IID, such as (1) IEEE-based-identifier incorporating EUI-64 address of the interface, (2) constant, semantically opaque IID, or (3) stable, semantically opaque IID;
  - a temporary IID [135].

   (c) For comparison, the computer also supports IPv4 and leases one IPv4 address from a DHCP server in the network.

2. The computer runs for more than eight days.

   (a) Temporary addresses are typically preferred for 24 hours. Hence, the computer regenerates a new temporary IID every 24 hours.

   (b) After 24 hours, the old temporary IID becomes deprecated; the IID is still assigned and used in sessions established when the IID was preferred. Temporary addresses are typically valid for seven days (preferred and deprecated combined).

3. Let us consider that the computer is restarted after several days (t_2). Alternatively, the computer can associate to a different Wi-Fi access point. The outcome regarding IIDs in use is similar.

   (a) The computer regenerates the same link-local IPv6 address.

   (b) The computer regenerates the same persistent IPv6 address (because the network address prefixes advertised in RAs are the same).

   (c) The computer generates a new temporary IID.

   (d) DHCP server continues to lease the same IPv4 address to the computer.

4. Following days, the computer keeps running and periodically generates new temporary IIDs (and drops the temporary IIDs older than seven days).

5. Suppose that the computer moves to a different LAN or that it associates to a different wireless network (t_3).

   (a) The computer regenerates the same link-local IPv6 address.

   (b) The computer learns the prefix used in the new network. For the prefix, the computer generates a new persistent and temporary address.

   (c) The computer leases a new IPv4 address.

6. Suppose that the computer returns to the original LAN or that it associates to the original wireless network (t_4):

   (a) The computer regenerates the same link-local IPv6 address.

   (b) The computer regenerates the same persistent IPv6 address as it originally configured in this network because the network address prefix advertised by ND is the same.

   (c) The computer generates a new temporary IID.

(d) DHCP server leases the computer the original IPv4 address or a new IPv4 address depending on its configuration and the time that elapsed after the computer disconnected from the original network.

As Figure 5.3 shows, IPv6 address assignment tracking needs to take into account that a single network interface can operate many IPv6 addresses at the same time, possibly tens of IPv6 addresses [35]. Additionally, the IPv6 addresses associated with the interface change over time. Hence, IPv6 address assignment tracking should incorporate time.

The Homenet architecture [29] describes a possible path of the IPv6 deployment in the future. One of the goals of the Homenet architecture is to provide [29, Section 3.2.4] multihoming to home networks. The Homenet architecture allows more globally routed prefixes to be used by a single device. Multipath TCP [68] splits a single application session into several subflows; each subflow can use a different IP address. Homenet architecture increases the number of IPv6 addresses on each interface — for each prefix, the computer configures at least one stable and one temporary IPv6 address. Each multipath TCP subflow can have a different global IPv6 address configured on an interface of the tracked computer.

In principle, the IPv6 LI is similar to IPv4 LI [28, Section 4.3]. The absence of NAT simplifies transport layer flows processing — two flows originating from the same IPv6 address at the same time belong to the same interface. Hence, it is straightforward to intercept data of the local computer (identified by one IPv6 address) [28, Section 4.3]. However, as a single interface can be configured with multiple IPv6 addresses that change over time, the challenge lays in the identification of all IPv6 addresses belonging to the same interface over time.

Identity graphs defined in this thesis in Chapter 8 tackles the challenge of multiple IPv6 addresses being configured to a single IPv6 interface by allowing an arbitrary number of IPv6 addresses to be linked to a single MAC address (or another unique interface identifier) during a particular period. The IPv6 address assignment tracking described in Chapter 6 is designed to learn all IPv6 addresses of all network interfaces on a LAN. The clock-skew-based identification studied in Chapter 7 discovers IPv6 addresses with similar clock skew which can be a sign that the addresses are being used by the same computer.

5.3 Dual-stack networks

Today, most of the web domain names resolve to IPv4 addresses only [1]. Therefore, most IPv6-enabled networks also support IPv4 to provide connectivity to IPv4 servers (typically through NAT or CGN). The simultaneous support for IPv4 and IPv6 is usually called dual stack. Hosts supporting both IPv4 and IPv6 create another challenge for LI: it is necessary to intercept traffic of both protocols.

Some content is accessible via both IPv4 and IPv6. A dual-stacked host can access the content via both protocols. Nevertheless, sometimes a network is misconfigured, sometimes one of the protocols experience higher latency or packet loss. Originally, IPv6 was preferred and only when the IPv6 connection did not succeed for several seconds, the host switched to IPv4. Later, *Happy Eyeballs* (HE) [194] allowed smooth fallback to IPv4.

HE can seamlessly switch between IPv4 and IPv6 without any user interaction. Therefore, a part of the content fetched from a single web server can be downloaded via IPv4 and the rest via IPv6. The multiplexing introduces an additional need for linking of IPv4 and IPv6 partial identities.

In addition, without HE, dual-stacked machines prefer IPv6. Hence, IPv6-enabled servers are accessed via IPv6 while for IPv4-only servers the host falls back to IPv4. As web pages often contain external content and DNS is accessed separately, one session may be carried over both IPv4 and IPv6 even without HE.

Moreover, multipath TCP [68] can multiplex an application session into several TCP subflows. Some subflows can utilise IPv4 while other subflows utilise IPv6.

Chapter 7 evaluates the clock-skew-based identification that estimates clock skew for both IPv4 and IPv6 addresses. Consequently, the identification mechanism can discover all IPv4 and IPv6 addresses of a single computer. Chapter 8 defines identity graphs that support dual stack networks.

5.4 Application layer protocols

LI standards include support of application layer protocols [61, 63]. The recent advent of smartphones introduces new proprietary protocols for instant messaging, voice communication (VoIP), and other forms of communication. Obviously, these applications can be misused by criminals. Therefore, a modern LI system and ISS has to allow support for new application layer protocols.

Typical mobile applications use a proprietary application layer protocol or a customised version of a standardised protocol [2, 12, 172]. Some applications use a telephone number as the user identifier, other applications use custom nicknames or other identifiers. Hence, the support of current mobile instant messaging requires the ability to support different protocols and different identifiers.

The support for application layer protocols creates several challenges for LI:

1. The number of application layer protocols is enormous, and it is continuously rising as new applications become available and protocols evolve. The huge variety of application layer protocols and the continuous protocol updates require that a modern LI system is extensible. Identity graphs described in Chapter 8 are extensible and already support several application layer identifiers. Nevertheless, due to time constraints, only a limited number of protocols is supported at the moment. I am currently working on the support for additional protocols as a part of the *Smart Application Aware Embedded Probes* (SProbe)[2] project.

2. Lately, many applications employ encryption by default. For LI systems, this means that application layer identifiers cannot be extracted directly from network links without encryption keys and online decryption. This thesis does not focus on encrypted protocols. Nevertheless, even data sent via the network in encrypted form are available decrypted at end hosts and often on the servers running the service. Identity graphs can be constructed based on different partial identity detectors that can be distributed across the network. Hence, the identity-related knowledge can be learnt from unencrypted sources, such as log files.

3. The variety of application layer protocols and the existence of many independent sources of identity information means that many partial identities of the same person (subject) exist. The existence of partial identities creates the need to link partial identities.

[2] http://www.fit.vutbr.cz/~ipolcak/grants.php?id=1003

Identity graphs presented in Chapter 8 are extensible and new partial identity detectors can be added easily. The built-in support for linking different partial identities based on network-related identifiers is prepared to support additional identifiers, including application layer identifiers. The identity graphs were already constructed [103, 159] from partial identity detectors parsing RADIUS [165], XMPP [169], IRC [106, 107], OSCAR (proprietary protocol for instant messaging), YMSG (proprietary protocol for instant messaging), SMTP [111], and HTTP [66].

5.5 Legal requirements

Based on the discussions with the Czech law enforcement agents, wordings of LI warrants vary. Some warrants allow interception of data identified by a specific IP address only. Other warrants demand interception of all traffic of a particular computer or a particular user. Some warrants allow linking identities whereas other warrants do not.

Another legal requirement concerns the covert nature of LI. The goal is to gather evidence for a court trial. Consequently, the interception has to be performed without the knowledge of the intercept target. Therefore, the location of the LI system has to take into account that is is not always possible to deploy its components into the most suitable location, such as the LAN where the intercept target is connected.

Another legal requirement is to differentiate between interception of IRI and CC [53]. For example [90, page 17], IRI intercepts allow monitoring of a suspect, linking his or her identity, and learning the communication parties of the suspect. An interception of the whole communication is legal only when the warrant allows interception of CC.

The IPv6 address assignment tracking defined in Chapter 6 create output symbols that can be converted to IRI records. Clock-skew-based identification can identify computers remotely; *pcf* can create messages that can be converted to IRI records. Identity graphs proposed in Chapter 8 can be created from partial identity detectors distributed anywhere on the Internet. The operations in identity graphs constraint the linking based on the wording of a warrant that orders an intercept.

5.6 The challenges and this thesis

As obvious from previous sections of this chapter, there are many challenges in LI in today networks. As Torres et al. [185] observed, an LI system of the future needs to process many partial identity detectors. Consequently, Chapter 8 proposes a highly distributed mechanism.

Figure 5.4 shows the problem decomposition employed by this thesis. The decomposition provides a basis for a modular, distributed and extensible IMS built from independent parts. Several partial identity detectors provide identity-related information from which an identity graph is automatically constructed. Each partial identity detector can be based on an independent mechanism. Chapters 6 and 7 describe two distinct partial identity detectors. Chapter 8 focuses on identity graphs that allow partial identity linking. Chapter 9 shows applications of the mechanisms studied during this PhD research.

The architecture depicted in Figure 5.4 tackles the challenges listed in this chapter by specific partial identity detectors focused on a specific source of identification. Each partial identity detector can be deployed in the most suitable location in the network. For example, the IPv6 address assignment tracking described in Chapter 6 (that addresses the challenge

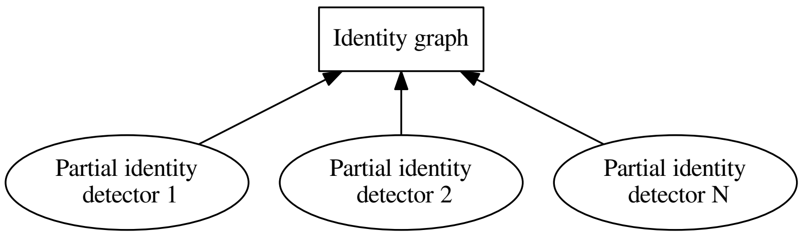

Identity graph
Partial identity detector 1
Partial identity detector 2
Partial identity detector N

Figure 5.4: The proposed IMS is modular, extensible and it can be distributed.

of multiple IPv6 addresses covered in Section 5.2) is deployable only directly on the LAN where the subject of the interception is located. Another example is a log processing system that parses application log files or Syslog [72] messages; in this case, the best deployment is the server that stores the log files. Consequently, the modular and extensible architecture is suitable to deal with different application protocols (see Section 5.4) as creating a new module does not require changes in other partial identity detectors.

However, it is not always possible to use a specific mechanism. For example, consider a court order to intercept data of a specific person. While it might be feasible to deploy the IPv6 address assignment tracking described in Chapter 6 to identify the intercept target in some locations, such as an international company, it is not possible to deploy the tracking in the home network of the subject as the interception would not be performed covertly — the subject is likely to recognize that an LI system was deployed in his house, see also Section 5.5. Hence, this thesis also considers remote identification.

To address the legal requirements that are presented in Section 5.5, this thesis focuses on both the local and remote identification. Figure 5.5 depicts the possible deployment of partial identity detectors. The benefits of local and remote detectors follow:

(a) User identification on LANs (see Figure 5.5 - a): The monitoring node is located on the same LAN as computers that are to be identified and intercepted. Hence, the learnt information is usually sound and complete.

(b) User identification outside the LAN of the intercept subject (see Figure 5.5 - b): Lately, users connect to the Internet using several Internet providers simultaneously (for example, using Wi-Fi and mobile carrier networks). Sometimes it is necessary to monitor the movement of intercept subjects that use different networks, such as public WiFi hotspots, hotel WiFi. As the deployment of an LI system in every network is not legally and technically possible, the remote monitoring is beneficial. The downside of remote identification is that some identifiers, such as MAC addresses, are not available and the identification may not be accurate.

For more details on local and remote identification, see also Section 3.4.

This thesis focuses in detail on two partial identity detectors:

1. Chapter 6 focuses on a local partial identity detection — IPv6 address assignment tracking on LANs. The proposed approach monitors ND messages on a LAN. The approach employs a timed transducer that tracks IPv6 address life-cycle of IPv6 addresses active on the LAN. The timed transducer is constructed based on the study of the behaviour of operating systems. The timed transducer detects that a new address was generated immediately after the transducer processes corresponding ND

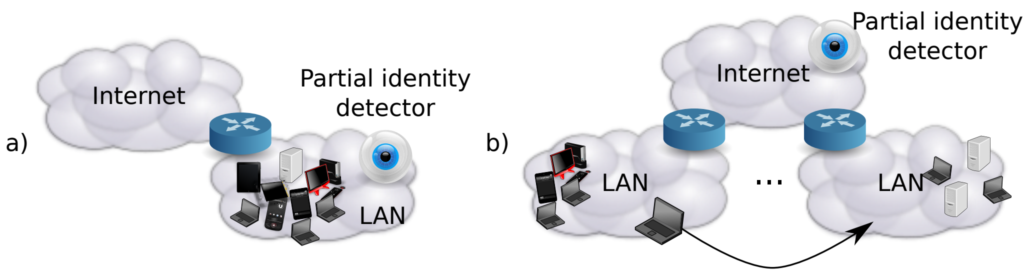


Figure 5.5: Local (a) and remote (b) deployment of partial identity detectors.

messages. Additionally, whenever an MLD querier queries a dropped address, the timed transducer detects that the address is not used anymore. The detection is passive for end devices and does not require any software or hardware modifications of switches and routers in the network. The method is suitable to detect all IPv6 addresses configured on a LAN, see Section 5.2.

2. The Clock-skew-based identification method studied in Chapter 7 provides remote identification. The method was reported to be fast [174] and handled passively [112]. During this PhD research, I extended the method to link all IPv4 and IPv6 addresses of a computer based on clock skew. Hence, the method addresses the challenges described in Sections 5.2 and 5.3. The method can also link traffic flows of the same computer hidden behind NAT [112], which deals with the challenge described in Section 5.1.

Besides these two partial identity detectors, Chapter 8 discusses additional partial identity detectors developed as a part of the Sec6Net project under my supervision [159].

Chapter 8 introduces identity graphs, a graph model that stores detected identifiers. Identity graphs link detected identifiers based on formally defined rules. The goal is to link identifiers on a warrant with dynamic identifiers that are the most suitable for identification of data to be intercepted, such as IP addresses [90, page 272]. Identity graphs support several types of linking constraints that were defined based on the possible network topologies and warrants wording. Identity graphs support multiple IP addresses of a single computer and the rules for identity graphs construction support NAT. Hence, Identity graphs address challenges described in Sections 5.1–5.5.

The methods presented in this thesis in Chapters 6 and 7 were included into SLIS [152, 159] developed as a part of the Sec6Net project. Chapter 9 describes SLIS and other applications of the methods developed during this PhD research.

Part II

Identity detection and linkage

Chapter 6

Identification on IPv6 LANs

This chapter focuses on the IPv6 address learning mechanism deployable on a LAN that is the main contribution of this thesis for accepting Hypothesis 1. The goal of the mechanism is to identify IPv6 and MAC address bindings on LANs, and consequently, identify all IPv6 addresses in use on all interfaces on a LAN. The mechanism can be deployed as a standalone solution that provides similar information to DHCP logs in IPv4, or, it can be utilised as one of the partial identity detectors for identity graphs described in Chapter 8.

The proposed IPv6 address assignment tracking mechanism examines the messages exchanged during ND and tracks the state of the discovered IPv6 addresses. For each IPv6 address, the mechanism learns the MAC address associated with the interface that uses the IPv6 address. The core of the mechanism is formally described as a timed transducer. The timed transducer executes transitions based on input symbols and time out events. The input symbols are constructed from messages received from the LAN; each symbol corresponds to a single message. The output of the transducer provides the address management information compatible with IRI *begin* and *end* records described in Chapter 4.

Firstly, we performed a study of current operating systems behaviour [149]. The study confirmed our expectations about several inconsistencies between ND implementations in different operating systems. Section 6.2 presents updated results of this study including the discovered inconsistencies.

Based on the study, we proposed [154] an extended finite state machine. The original paper [154] was selected for an extended version that was published as *Host Identity Detection in IPv6 Networks* [156]. I was the main author of the papers; I wrote the text, the ND study was conducted under my supervision. In this thesis, I transformed the vague description of the extended finite state machine and defined the mechanism as a timed transducer. The timed transducer is defined in Section 6.4. As a part of the Sec6Net project, under my supervision, we developed a tool called *ndtrack* [93] that implements the IPv6 address assignment tracking mechanism.

6.1 Related work in IPv6 identification

Section 3.3 introduces IPv6 addressing, IIDs, and ND including SLAAC. SLAAC is a mandatory address assignment method during which a host generates one or more IPv6 addresses by appending a custom IID to an advertised prefix. SLAAC brings new challenges for network operators [81][1]. This section provides an overview of methods for local identification in IPv6 networks.

6.1.1 Local monitoring of address assignments by neighbor cache polling

All IPv6-enabled hosts maintain a *neighbor cache* (NC). NC is a table that stores the bindings of IPv6 and MAC addresses of computers, with which the host has active or recently finished communication on a LAN. See Figure 6.1 for an example of an NC of a Cisco router.

```
SLI#sh ipv6 neighbors
IPv6 Address                       Age Link-layer Addr State Interface
2001:DB8::1AA9:5FF:FE90:1540         0 18a9.0590.1540  REACH Gi0/0
2001:DB8::20B:82FF:FE3A:F957         0 000b.823a.f957  REACH Gi0/0
2001:DB8::3E97:EFF:FEDB:F029        13 3c97.0edb.f029  STALE Gi0/0
2001:DB8::A00:27FF:FE16:1D55         0 0800.2716.1d55  REACH Gi0/0
2001:DB8::92E6:BAFF:FE83:9148        0 90e6.ba83.9148  REACH Gi0/0
2001:DB8::204:96FF:FE1D:4E30        15 0004.961d.4e30  STALE Gi0/0
2001:DB8::F6EC:38FF:FEF0:873B        0 f4ec.38f0.873a  REACH Gi0/0
```

Figure 6.1: An example of the neighbor cache of a Cisco router.

Grégr et al. [81] poll routers in the Brno University of Technology network and download NC of the routers in the University network to learn the IPv6 addresses of connected end hosts in the network. However, the polling increases the workload carried by the routers. Hence, the polling cannot be too frequent as it would result in negative performance impact. Grégr et al. are not interested in immediate detection of a newly assigned IPv6 address. Instead, their use case allows a long polling interval, for example, 15 minutes or even 1 hour. Such delay is not acceptable for LI.

In addition, an NC stores information about directly connected nodes only. Hence, in network topologies with multiple routers that segment a network into separate subnetworks, the IPv6 address monitoring has to gather information from all routers that are connected to segments with end user devices as depicted in Figure 6.2 (a).

Virtualization, such as Cisco Virtual Switching Systems [31] or HP Intelligent Resilient Framework [84], can prevent the need to poll multiple routers. The virtualization lets several devices act as a single device. Consequently, the virtualized devices share common NC, and it is sufficient to probe only one physical device. For example, see Figure 6.2 (b).

ND snooping tracks IPv6 addresses configured to devices attached to a switch on the access layer. When enabled, ND snooping creates a table similar to the NC of routers. Therefore, it is possible to poll access layer switches instead of the routers. However, ND snooping is not supported on all switches, and the additional workload is more critical in comparison to the NC polling as access layer switches tend to have lower resources.

[1] Requests for IPv6 address tracking reappear on the Internet, for example, https://code.google.com/p/android/issues/detail?id=32621#c60 or https://www.ietf.org/mail-archive/web/v6ops/current/msg22650.html

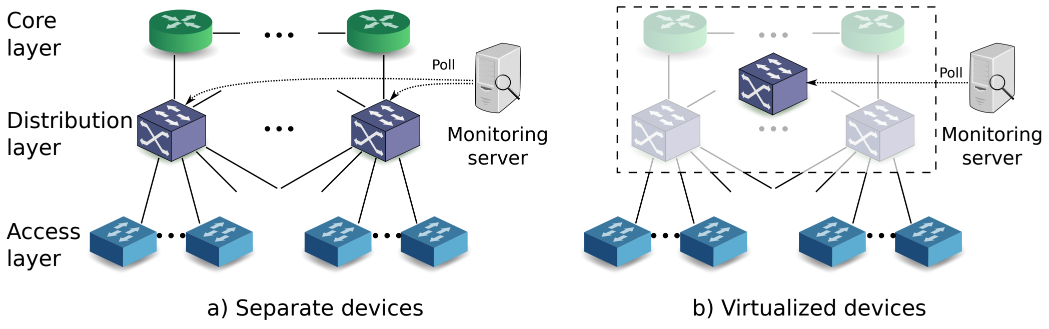


Figure 6.2: NC polling in a) traditional networks, b) networks with virtualized devices.

Compared to the router or switches polling method [81], this chapter focuses on a method that:

- detects new addresses immediately, including addresses that are utilised for intra-LAN communication only,
- does not poll routers or switches and thus does not increase their load periodically,
- works in networks with switches without NC and MLD snooping (for example, low-priced switches commonly deployed in the access layer),
- detects that addresses are released by end hosts even during the time when the addresses are still present in the NC of a router,
- is deployable in networks without virtualized switches [31, 84].

Asati and Wing [8] tried to change router behaviour so that routers propagate changes in NC through DHCPv6 messages to a DHCPv6 server. The DHCP server would store such information. Consequently, DHCPv6 log files would provide information about all IPv6 addresses used in the network. However, their work did not reach the publication as RFC. Therefore, it is unlikely that the mechanism is implemented in routers.

6.1.2 DHCPv6 tracking

As stated earlier, DHCPv6 is an optional method to assign IPv6 addresses to end hosts. Hence, an IPv6-enabled LI system should monitor address assignments through DHCPv6.

Groat et al. [78] studied DHCPv6 for monitoring the identity of users on LANs. They focused on possibilities of an adversary that has access to several IPv6 LANs. The adversary sniffs and spoofs DHCPv6 messages. An adversary can operate:

1. Directly on the LAN of the tracked device – the adversary monitors DHCPv6 traffic.
2. On the LAN of the DHCPv6 server – the adversary can sniff the DHCPv6 traffic or access the database of the leases stored on the server.

Nevertheless, Groat et al. [78] focus only on one particular address assignment method — DHCPv6. SLAAC is the default address assignment method in IPv6 whereas DHCPv6 is optional. Moreover, even if DHCPv6 is deployed, end hosts can still employ SLAAC to generate additional IPv6 addresses provided that the RA contains prefix information option with the autonomous address-configuration bit set [136]. As DAD must verify the uniqueness of an IPv6 address assigned by a DHCPv6 server, the IPv6 address assignment tracking proposed in this chapter also detects DHCPv6 leases. Moreover, identity graphs proposed in Chapter 8 support DHCPv6 as a partial identity detector; a module for DHCPv6 lease tracking was developed as a part of the Sec6Net project [159]. The module provides information required to build identity graphs.

6.1.3 Web-based dual stack address discovery

Sanguanpong et al. [170] implemented a captive portal for dual-stacked networks. When a host connects to the Kasetsart University network, the user needs to authenticate to the network. The first HTTP request is redirected to the captive portal where the user can enter his or her credentials.

The authentication web page embeds two images, one accessible through IPv4 and one through IPv6. Both images are identified with a unique hash. When the browser downloads the image, it provides the unique hash to the web server. Consequently, the web server links the IPv4 address and an IPv6 address by the hash supplied in both HTTP requests.

The downside is that the method discovers only one IPv4 and one IPv6 address. In IPv6, each device has at least one link-local and one routable IPv6 address. Usually, the device generates more than one routable address, each with a different type of IID (see Section 5.2 for more details). In the case of a temporary IID, the device regenerates a new IID periodically.

The web-based dual stack address discovery reveals only the address that is preferred for the communication with the web server at the time of the access to the captive portal. Consequently, the method is not suitable for LI since LI needs to cover all traffic of the suspect. Moreover, the identification for LI has to be transparent to the suspect. The detection by the proposed timed transducer is transparent to hosts. In contrast, the captive portal visibly modifies the behaviour of the network.

6.1.4 Neighbor Discovery tracking

During SLAAC [184], hosts in IPv6 network utilise ND messages [136] to learn on link IPv6 addresses, the prefix for IPv6 address autoconfiguration, and check that the IPv6 address to be assigned is not already used in the network. Section 3.3 provides more information about the procedure.

The messages exchanged during ND can be monitored by *addrwatch* [114]. Figure 6.3 provides an example *addrwatch* output. Each line represents a single ND message. Each line displays the timestamp of the message, the network interface that captured the message, the VLAN number, the MAC address, the IPv6 address and the ND message type. Hence the tool gives the network operator an overview about the ND traffic, the tool can be used to log IP address assignment in the network.

```
1329486484 eth0 0 00:aa:bb:cc:dd:ee fe80::2aa:bbff:fecc:ddee ND_NS
1329486486 eth0 7 00:11:11:11:11:11 fe80::211:11ff:fe11:1111 ND_NS
1329486487 eth0 7 00:22:22:22:22:22 fe80::222:22ff:fe22:2222 ND_DAD
```

Figure 6.3: An example of the *addrwatch* output.

Although *addrwatch* displays ND messages captured from the network, it does not display the state of the IPv6 addresses. Hence, by looking at the output, it is not clear, what IP addresses are active on the network, what interface has currently assigned an IPv6 address A and if it is the same interface that had the IPv6 address A assigned at a particular time. Additionally, *addrwatch* does not track the lifetime of IPv6 addresses. Consequently, *addrwatch* does not display any line when an IPv6 address expires.

The timed transducer defined in Section 6.4 also tracks ND messages. In contrast to *addrwatch*, the timed transducer keeps information about the state of an IPv6 address and the MAC address of the interface that has assigned the IPv6 address. Additionally, the timed transitions allow the timed transducer to automatically track the tentative and valid period of an IPv6 address. The implementation of the timed transducer can output similar information to *addrwatch* including the timestamp of a new and expired IPv6 address assignment.

6.1.5 Other host discovery methods applicable in local IPv6 networks

A passive traffic analyser can monitor all traffic on a LAN, for example, by copying all traffic with a Switch Port Analyser[2] or a tap. If the analysis happens on all links, such analysis detects all local addresses. Nevertheless, the passive analysis of local traffic requires much more computing power compared to the analysis by timed transducers described in Section 6.4.

An active adversary can *ping* multicast groups such as the all host multicast group *ff02::1*. However, some operating systems do not reply to these *ping* requests [75]. An alternative solution is to send invalid or unspecified options in the packets [75]. Nevertheless, each host typically replies from a link-local address. The IPv6 address assignment tracking described in this chapter is passive for monitored devices and detects all IPv6 addresses of the hosts.

6.2 Study of Neighbor Discovery implementations

For LI, it is necessary to deploy a mechanism that detects a new IPv6 address assignment as fast as possible. Hence, we decided that the mechanism should observe ND, DAD in particular. However, we realised that the message sequence exchanged during DAD is not the same for different operating systems; we observed *protocol obscurities* [65, Section 2.1] in ND implementations. We decided to study [149] the exact sequences of messages during DAD created by several operating systems. This section provides an updated version of the study. The study focuses on different types of IPv6 addresses (SLAAC-generated, DHCPv6 leases, static). The study discovers the differences between the defined message sequence [184] and current implementations. The study proves that differences do exist.

This section describes the implementation of ND in different operating systems. The study focuses on Windows, Linux, Mac OS X, FreeBSD, OpenBSD, and Solaris. Table 6.1 shows the exact version of studied operating systems for each family. In addition to the original tests [149], for the purpose of this thesis, I rerun the tests with recent versions of Linux (kernel 3.19), Mac OS X 10.7.5, and Windows 10. The results of this study influenced the proposed timed transducer that detects IPv6 addresses on a LAN as described in Subsection 6.2.4.

[2] https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3560/software/release/12-2_52_se/configuration/guide/3560scg/swspan.html

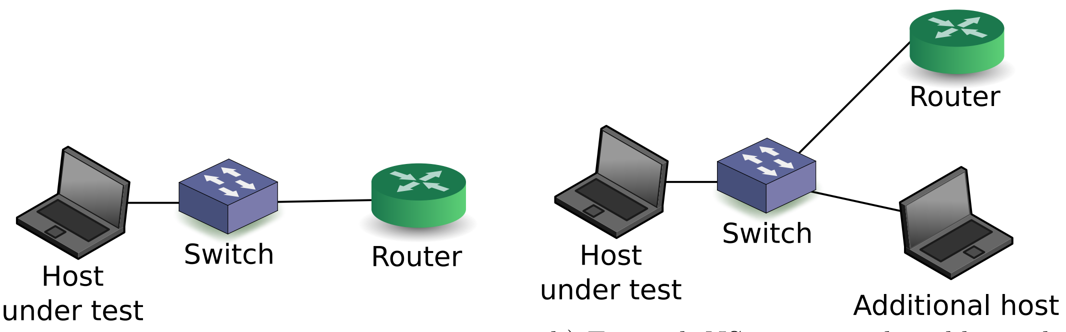
Table 6.1: Operating systems tested for compliance during ND.

Windows	XP SP3, Vista, Vista SP3, Server 2008 R2, 7, 7 SP1, 8, and 10
Linux	kernel 2.4.27: Debian 3.1
	kernel 2.6.18: Red Hat 5
	kernel 2.6.32: CentOs 6.2, Debian 6.0.4, Ubuntu 10.04
	kernel 2.6.38: Mandriva
	kernel 3.0: Linux Mint 12, Ubuntu 11.10
	kernel 3.1: Fedora 16
	kernel 3.2: Ubuntu 12.04
	kernel 3.19: Ubuntu Server 15.04
Mac OS X	10.6.2 (kernel 10.2), 10.7.5 (kernel 11.4.2)
Unix	FreeBSD 9.0, OpenBSD 5.0, and Solaris 5.11

6.2.1 SLAAC message sequence

Firstly, we tested basic SLAAC implementation. We observed that operating systems join multicast groups, perform DAD, and do not generate addresses that are already present in the network.

Figure 6.4 depicts the network used for DAD testing. We were interested in both successful and failed address assignments. To test the former, we used the network (a); to test the latter, we connected an additional computer (b) that injected the NA messages that forced the host under test not to use specific addresses.



a) All address assignments were successful in this network.

b) For each NS message, the additional host in the network replied with an NA pretending that it uses all addresses.

Figure 6.4: Network topologies used for SLAAC testing.

For successful address assignments, we designed the following scenario:

1. The router and the switch were running. The router was configured to send RAs with a custom prefix. The interface of the host under test was not active.

2. The interface of the host under test was activated.

    - The host configured a link-local address.
    - The host configured one IPv6 addresses or more. In all operating systems, one address had an IEEE-identifier-based IID or a constant, semantically opaque IID. Some operating systems configured additional addresses, for example, temporary IIDs [135].

We expected the following behaviour that is based on RFC 4862 [184, Section 5]:

1. The host under test generates an IID for the link-local address, joins the appropriate solicited-node multicast group [89], and performs DAD.
2. The host sends an RS, the router replies with an RA, and the host learns the prefix configured for the test network.
3. For each additional address to be configured, the host:
   (a) generates the IID.
   (b) joins the appropriate solicited-node multicast group [89] (if the host is not already present in the group).
   (c) performs DAD for the address generated for the IID.
4. As the router periodically queries multicast groups, the host confirms its subscriptions.

Nevertheless, the DAD procedure can be slightly modified with extensions such as the optimistic DAD [127]. The goal of the study was to learn what are the real implementations of DAD.

The expected outcome was that the host generates IPv6 addresses. All tested operating systems were successful. During testing, we observed the following behaviour:

- Linux machines follow the original message sequence [184]: for each address A, the host joins the solicited-node multicast group, performs DAD, and only when no other host replies, the host uses the address as a source address A.
- MAC OS X machines also follow the original message sequence [184]. Nevertheless, for each newly assigned address, MAC OS X sends an unsolicited NA with the override flag set as described below.
- Windows Vista and later use an optimistic link-local address A [127] to join multicast groups before they start DAD for the address A.
- Windows Server 2008 R2, Windows 7 and later send NS-DAD for link-local addresses before they subscribe to the solicited-node multicast group corresponding to the tentative address.
- Solaris (for all addresses) sends NS-DAD before it subscribes to the solicited-node multicast group corresponding to the tentative address.
- OpenBSD does not join solicited-node multicast groups at all.
- Windows 8 and later, Solaris, and recent Mac OS X (for example, 10.7.5) advertise their presence by unsolicited NAs during DAD (that are not mandatory) to all hosts in the network (to the ff02::1 multicast group). Windows 8 and Mac OS X set the override flag; they try to update invalid NC entries (for example, to override entries created by another host using the same address recently).
- All operating systems that subscribe to multicast groups reply to MLD queries periodically sent by the router. However, in some cases, we observed that the maximal timeout specified in the RAs was not met. Although RAs specified the maximal timeout of 0.1 second, some replies to the MLD queries were received after 0.7–0.8 seconds.

To test the collision detection behaviour of the operating systems, we connected an additional host to the network (see Figure 6.4 b). We tested the following scenario:

1. The router, the switch, and the additional host were running. The router was configured to send RAs with a custom prefix. The interface of the host under test was not active.
2. There was an additional host in the network.
   (a) The additional host was either configured statically with the same address as the host under test (this is possible in the case of predictable IIDs), or,
   (b) the additional host simulated collisions of unpredictable IPv6 addresses with a customised script that replied to all NS-DADs with a fake NA; effectively, the additional host was pretending that it has each address already assigned.
3. The interface of the host under test was activated. Then, the host tried to configure IPv6 addresses similarly to the previous scenario.

We repeated the scenario and configured the additional host to cause collisions only for a specific case in each run. The expected outcome was that the operating systems do not use the colliding address. The testing was successful except the test where we configured both hosts with the same MAC address. In this case, FreeBSD and all versions of Windows except Windows XP ignored the NA with the same MAC address as their own.

The results show that some operating systems do not follow the message sequence described in RFC4862 [184] and RFC 4429 [127]. The impact on the IPv6 address assignment tracking proposed in this chapter is summarised in Subsection 6.2.4.

6.2.2 DHCPv6 support and message sequence

RFC 3315 [19, paragraph 18.1.8] recommends that a DHCPv6 client that is about to utilise the assigned address for communication should perform DAD. Additionally, RFC 4862 [184, Subsection 5.4] requires that DAD must be performed on all unicast addresses. As the IPv6 address assignment tracking proposed in Section 6.4 monitors DAD, we wanted to test real implementations. For the ND test triggered by DHCPv6, we connected a DHCPv6 server to the network and reconfigured the router. The network is depicted in Figure 6.5. The main goal is to study DAD for DHCPv6-leased addresses.

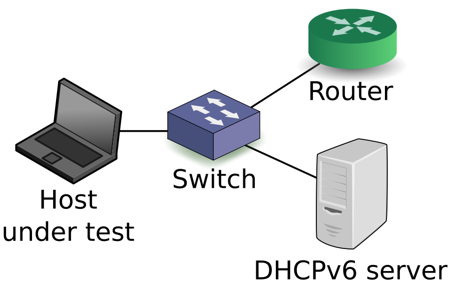


Figure 6.5: The network for monitoring the interaction between SLAAC and DHCPv6

We tested Linux (Ubuntu 12.10 and Mandriva 2011), Solaris 11.11, Windows 7, 8 and 2008 R2. The test scenario follows.

1. The router, the switch, and the DHCPv6 server were running. The router was configured to send RAs with the flag signalling that a DHCPv6 server is present in the network. The interface of the host under test was not active.

2. The interface of the host under test was activated.

   - The host configured a link-local address.
   - The host learnt about the presence of the DHCPv6 server from an RA.
   - The host leased an IPv6 address from the DHCPv6 server.
   - The host periodically refreshed the lease.

All operating systems performed DAD when they leased a new IPv6 address as expected. However, the behaviour diverged during the lease refresh:

- Linux distributions did not perform DAD during lease refresh.

- Solaris repeated DAD while it resubscribed to the solicited-node multicast group corresponding to the leased address during each lease refresh.

- All versions of Windows repeated DAD during each lease refresh, but they did not explicitly resubscribe to the solicited-node multicast group corresponding to the leased address.

RFCs do not mandate the repeated DADs. Hence, all tested operating systems behaved in conformance with the expectations and followed the advice by RFC 3315 [19, Paragraph 18.1.8] and requirement in RFC 4862 [184, Subsection 5.4]. Consequently, the IPv6 address assignment tracking proposed 6.4 tracks also DHCPv6-leased IPv6 addresses.

6.2.3 Duplicate Address Detection for static address assignments

The requirement to perform DAD for all unicast addresses [184, Subsection 5.4] also holds for static addresses. To validate that DAD occurs in practice, we run the final scenario focused on DAD for static addresses. Figure 6.6 depicts the network.

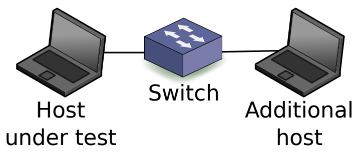


Figure 6.6: The network used for static address assignments testing.

We tested both unique and colliding address assignments. The scenario was as follows:

1. Both hosts in the network were running. The additional host was preconfigured with static IPv6 addresses and a link-local IPv6 address.

2. We configured a new IPv6 address on the host under test that did not collide with any address on the additional host.

3. We configured a new IPv6 address on the host under test that collided with one of the IPv6 addresses preconfigured on the additional host.

The expected behaviour was that the unique address was assigned to the interface and the colliding address was dropped. We identified the following observations:

1. Solaris and FreeBSD send NS-DAD before they join the solicited-node multicast group corresponding to the tentative address.
2. Windows 8, Solaris, and recent Mac OS X (for example, 10.7.5) advertise their presence by unsolicited NAs during DAD (which are not mandatory) to all hosts in the network (the ff02::1 multicast group). See the explanation in Subsection 6.2.1.
3. OpenBSD does not join solicited-node multicast groups.

The results were similar to those collected during the tests of SLAAC (see Subsection 6.2.1). The main difference is that during SLAAC, FreeBSD first subscribes to the solicited-node multicast group and later sends NS-DAD during SLAAC, the order is the opposite for the static address assignments (NS-DAD followed by the MLD report that joins the node to the solicited-node multicast group).

6.2.4 Summary of observations of ND implementations

The study of ND implementations in current operating systems reveals that there are some differences. Nevertheless, it is possible to provide a mechanism that is compatible with the majority of operating systems.

Figure 6.7 depicts the life cycle of an IPv6 address as defined by RFC 4862 [184] as observed from the position of the host that generates the IPv6 address. The address can be used for communication in the *preferred* and the *deprecated* state. A monitoring device has to detect the transitions between (1) the *tentative* and the *preferred* state, (2) the *preferred* and the *invalid* state, and (3) the *deprecated* and the *invalid* state. These transitions change the availability of the address for communication of the monitored host.

Linux hosts follow the standard sequence [184] most closely. A monitoring of IPv6 address management traffic of Linux hosts can detect:

1. A change between the *invalid* and the *tentative* state by observing MLD reports joining a *solicited-node multicast group*, and the following NS-DAD in the *solicited-node multicast group*.
2. A change between the *tentative* and the *preferred* state when there is no reply for the NS-DAD for a period defined by *RetransTimer* [136].
3. A change between the *deprecated* and the *invalid* state can be computed from the valid lifetime contained in RAs [136]. Alternatively, in the presence of an MLD querier, the address is invalid when there is no reply for an MLD query to the *solicited-node multicast group*.
4. The change between the *preferred* and the *deprecated* state can be computed from the preferred lifetime contained in RAs [136]. However, this is not necessary for the timed transducer proposed in Section 6.4 as the address can be used for communication in both states.

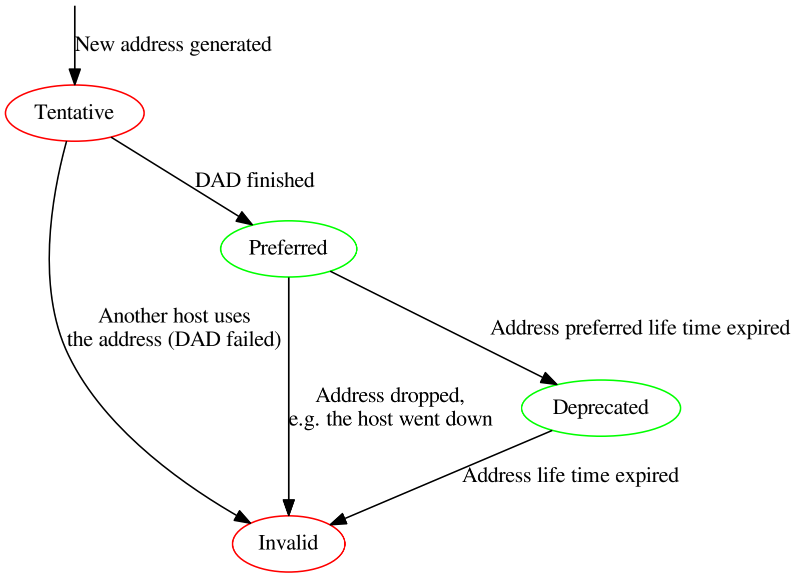

New address generated
Tentative
DAD finished
Preferred
Another host uses the address (DAD failed)
Address preferred life time expired
Address dropped, e.g. the host went down
Deprecated
Address life time expired
Invalid

Figure 6.7: IPv6 address life cycle as defined by RFC 4862 [184].

The same messages can be observed by monitoring Apple Mac OS X. Additionally, a change from the *tentative* to the *preferred* state of a Mac OS X host can be detected from an unsolicited NA issued during DAD.

FreeBSD follows the standard as well, so it is possible to detect the transitions between (1) the *invalid* and the *tentative* state, (2) the *tentative* and *preferred* state, and (3) the *deprecated* and *invalid* state. The only exception is a static address for which it may not be possible to detect the change between the *invalid* and the *tentative* state as FreeBSD sends NS-DAD before it joins the *solicited-node multicast group*.

Windows Server 2008 R2, Windows 7 and later send an NS-DAD before they subscribe to the *solicited-node multicast group* for the address. However, it is possible to detect the change from the *tentative* to the *preferred* state from the additional unsolicited NAs.

Solaris sends NS-DAD before it subscribes to the *solicited-node multicast group*. Hence, the change from the *invalid* state to the *tentative* state may not be detected. However, as it issues unsolicited NAs during DAD, a monitoring node can detect the change from the *tentative* state to the *preferred* state.

OpenBSD does not join *solicited-node multicast groups*. Therefore, it is not possible to observe NS-DADs in its tentative phase unless they are broadcasted by the network (no MLD snooping). We decided that the market share of the operating system is too low and the timed transducer proposed in Section 6.4 does not detect OpenBSD in networks with MLD snooping.

As a result, we decided to construct timed transducer. The input symbols of the timed transducer represent IPv6 address management messages discussed in this section. The output symbols of the timed transducer contain information about the assignments of an

IPv6 address. The goal is to detect the change from the *invalid* state to the *tentative* state, and consequently, the change from *tentative* state to the *preferred* state as described above for a Linux host. Additionally, the proposed IPv6 address assignment tracking based on timed transducer utilises unsolicited NAs to detect that an unobserved address shifted to the *preferred* state. The timed transducer should delay transitions from both the *preferred* and the *deprecated* states to the *invalid* state as we observed that some replies to MLD queries arrived after 0.7–0.8 seconds instead of the advertised 0.1 second,

6.3 Theoretical background for proposed IPv6 address assignments tracking

This section provides the mathematical background for the timed transducer proposed for the local IPv6 address assignment tracking.

Let ϵ denote an empty string in the following text.

As resolved in Section 6.2, the proposed IPv6 address assignment tracking is based on a timed transducer. Definition 6.1 defines Mealy machine — a deterministic finite state transducer without a notion of time.

Definition 6.1. A **Mealy machine** is a 5-tuple $(S, S_0, \Sigma, \Lambda, \delta)$, where:

- S is a finite set of states,
- $S_0 \in S$ is an initial state,
- Σ is a finite input alphabet ($\Sigma \cap S = \emptyset$, $\epsilon \notin \Sigma$),
- Λ is a finite output alphabet ($\Lambda \cap S = \emptyset$, $\epsilon \notin \Lambda$),
- $\delta \colon S \times \Sigma \to S \times \Lambda^*$ is a transition function mapping a pair of a state and an input symbol to a pair of a new state and an output sequence (given a specific input symbol, δ shifts the machine from one state to another while it produces an output).

Alur and Dill [6] consider the *dense-time* model, in which time is a dense set. The time is represented with real numbers that increase monotonically without bounds. Clock variables can be compared with clock constraints as defined by Alur and Dill [6] in Definition 6.2.

Definition 6.2. For a set X of clock variables, the set $\Phi(X)$ of **clock constraints** δ is defined inductively by

$\delta \equiv x \leq c | c \leq x | \neg\delta | \delta_1 \wedge \delta_2$,

where $x \in X$ is a clock from the set X and $c \in \mathbb{Q}$ is a constant.

Alur and Dill [6] also define abbreviations of clock constraints, such as **true**, which means that there is no clock constraint. This thesis follows their notation.

Definition 6.3 defines a timed transition table [6]:

Definition 6.3. A **timed transition table** is a 5-tuple (Σ, S, S_0, C, E), where:

- Σ is a finite alphabet,
- S is a finite set of states,
- $S_0 \subseteq S$ is a set of start states,

- C is a finite state of clocks,
- $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ gives the set of transitions. An edge $(s, s', a, \lambda, \phi)$ represents a transition from state $s \in S$ to state $s' \in S$ on input symbol $a \in \Sigma$. The set $\lambda \subseteq C$ gives the clocks to be reset with this transition. ϕ is a clock constraint over C.

Let us combine the Mealy machine (Definition 6.1) and timed transition table (Definition 6.3) into a timed transducer in Definition 6.4.

Definition 6.4. A **timed transducer** is a 6-tuple $(S, S_0, \Sigma, \Lambda, C, \delta)$, where:

- S is a finite set of states,
- $S_0 \in S$ is an initial state,
- Σ is a finite input alphabet ($\Sigma \cap S = \emptyset$, $\epsilon \notin \Sigma$),
- Λ is a finite output alphabet ($\Lambda \cap S = \emptyset$, $\epsilon \notin \Lambda$),
- C is a finite state of clocks,
- $\delta \colon S \times (\Sigma \cup \epsilon) \times \Phi(C) \to S \times \Lambda^* \times 2^C$ is a transition function mapping a triplet of a state, an input symbol (or empty string), and a clock constraint over C to a triplet of a new state, an output sequence, and a set of clocks to be reset. It means, given a specific input symbol, δ shifts the timed transducer from one state to another while it produces an output if and only if the specified clock constraint hold.

Note that Definition 6.4 allows shifting without reading an input symbol.

6.4 Proposed IPv6 address assignments tracking

Based on the study of the DAD implementation in different operating systems described in Section 6.2, we proposed [154, 156] and implemented [93] the address assignment tracking on IPv6 LANs.

This section formally describes the core of the proposed tracking mechanism as a set of timed transducers (see Definition 6.4).

A single proposed timed transducer tracks a single IPv6 address A. Informally, the proposed timed transducer stores the MAC address identifying the interface to which IPv6 address A is being assigned in the states of the timed transducer. At the input, the proposed timed transducer reads symbols constructed from NS-DAD messages related to the IPv6 address A, NA messages of A, MLD queries for the solicited-node multicast group corresponding to A, and MLD reports (replies) for the queries. The timed transducer outputs which MAC address is linked to the IPv6 address A and which MAC address is no longer linked to the IPv6 address A.

As already mentioned, a single timed transducer described in this section tracks only a single IPv6 address A. To track all IPv6 addresses in the network, one timed transducer is required for each IPv6 address. The need for a vast number of automata is reduced in practice. It is sufficient to store and track only addresses that are not in the initial state. Hence, there is not any substantial memory overhead.

The proposed timed transducer is executed on a monitoring node as depicted in Figure 6.8. The monitoring node is connected to any switch in a network as any other host.

To ensure that the multicast ND traffic is delivered to the monitoring node, the node has to join the multicast group for all nodes (*ff02::1*), all MLDv2-capable routers (*ff02::16*), and all solicited-node multicast groups detected by analysing MLD reports in the multicast group for all MLDv2-capable routers. A symbol converter transforms the IPv6 address management messages in these multicast groups to the input symbols of the timed transducer. The proposed timed transducer creates the output that is available for processing by other software, for example, to construct identity graphs, see Chapter 8.

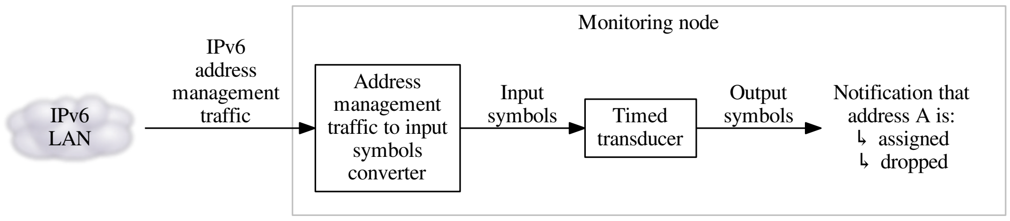

Monitoring node
IPv6 LAN
IPv6 address management traffic
Address management traffic to input symbols converter
Input symbols
Timed transducer
Output symbols
Notification that address A is:
↳ assigned
↳ dropped

Figure 6.8: Monitoring node observes address management traffic in the network. Besides the proposed timed transducer, the IPv6 address assignment tracking incorporates a converter that transforms network messages to the input symbols of the proposed timed transducer.

A monitoring node in a network cannot accurately detect all events and state changes happening on network nodes [17]. Therefore, the proposed timed transducer focuses on detecting substantial states of address assignments. Specifically, it detects that an IPv6 address (1) is not used in the network, (2) is in the tentative state, or, (3) is considered preferred or valid by a host in the network.

Before we formally define the proposed timed transducer, let us define the following finite sets that are later used to construct the tracking time transducer:

- the set of all MAC address values V_{MAC} [181],
- the set of all IPv6 address values V_{IPv6} [89],
- the set of all IPv6 multicast addresses $V_{\text{IPv6-M}}$ (*ff00::/8* [89]).

Following subsections define the components of the proposed timed transducer $\mathcal{M} = (S, S_0, \Sigma, \Lambda, C, \delta)$. Equivalence 6.1 defines S, Equivalence 6.2 defines S_0, Equivalence 6.7 defines Σ, Equivalence 6.10 defines Λ, Equivalence 6.11 defines C, and Equivalence 6.27 defines δ.

6.4.1 States of the timed transducer $\mathcal{M}$

An IPv6 address goes through several phases during its lifetime [184] (see also Figure 6.7). Under typical conditions, an address first reaches the tentative state (during which the node validates that the address is not already active), then the address reaches the preferred state, later the address becomes deprecated, and finally, it becomes inactive. Alternatively, instead of being tentative, the host can optimistically use the address for communications before it validates its uniqueness [127]. However, the node should avoid using an address in the optimistic state unless necessary and the node cannot announce the possession of the address in NS messages.

Any IPv6 tracking mechanism based on ND messages needs to take the IPv6 address states into account. An *optimistic*, *preferred*, or *deprecated* address can be used for communication. A tentative address is tentatively assigned to an interface but it is not used until its uniqueness is verified.

The proposed timed transducer differentiates between the following monitoring states for an address:

Address not used (*NotUsed*) meaning there is no known address assignment of IPv6 address A.

Address in DAD (*InDAD*) meaning there is a host in the network performing DAD for the address — the host treats the address as *tentative* or *optimistic*.

Address assigned (*Assigned*) meaning there is a host in the network that treats the IPv6 address as *preferred* or *deprecated*.

Assignment being validated (*BeingValidated*) meaning that an MLD querier validates active multicast groups in the network.

Another host in DAD (*AnotherDAD*) resolves circumstances during which another host performs DAD.

Let us define the set $S_E \equiv \{\text{NotUsed}, \text{InDAD}, \text{Assigned}, \text{BeingValidated}, \text{AnotherDAD}\}$ as the set of IPv6 address tracking states. Additionally, let us define the set $S_M \equiv \{\text{InDAD}, \text{Assigned}, \text{BeingValidated}\}$.

The states S of the proposed timed transducer $\mathcal{M}$ are defined as

$$S \equiv S_M \times V_{\text{MAC}} \cup \{(\text{NotUsed}, \epsilon)\} \cup \{\text{AnotherDAD}\} \times V_{\text{MAC}} \times V_{\text{MAC}}. \tag{6.1}$$

The components of the states in set S encode:

- For tracking states in S_M, S contains pairs compound of two elements: (1) the expected state of the tracked IPv6 address A, and (2) the MAC address of the interface having the IPv6 address A assigned.
- In the *NotUsed* tracking state, there is no linked MAC address.
- For the *AnotherDAD* tracking state, the timed transducer $\mathcal{M}$ needs to remember both the MAC address of the original host and the MAC address of the another host performing DAD.

The initial state of the timed transducer $\mathcal{M}$ is

$$S_0 \equiv (\text{NotUsed}, \epsilon). \tag{6.2}$$

6.4.2 Alphabets of the timed transducer $\mathcal{M}$

The timed transducer $\mathcal{M}$ tracks ND messages, and MLD queries and reports (responses). The timed transducer $\mathcal{M}$ outputs information about an address being active (the assignment begins or ends).

Let $s^{(a,A)}$ denote an NS-DAD querying the presence of the IPv6 address A on another host in the network by a computer with a MAC address $a \in V_{\text{MAC}}$. Let us define the set

$V^A_{\mathrm{NS-DAD}}$ containing symbols for all possible NS-DAD queries for the IPv6 address A in Equivalence 6.3. The set $V^A_{\mathrm{NS-DAD}}$ contains one symbol for each valid MAC address.

$$V^A_{\mathrm{NS-DAD}} \equiv \left\{ s^{(a,A)} : a \in V_{\mathrm{MAC}} \right\} \tag{6.3}$$

Whenever the monitoring node running the timed transducer $\mathcal{M}$ intercepts an NS-DAD for the address A, the input symbol converter transforms the message into the symbol $s^{(a,A)}$. The symbol refers to both the IPv6 address A and the MAC address a.

Additionally, let $n^{(a,A)}$ denote an NA advertising that an interface identified by the MAC address $a \in V_{\mathrm{MAC}}$ has the IPv6 address A assigned. Let us define the set V^A_{NA} representing all possible NAs for the IPv6 address A in Equivalence 6.4. The set V^A_{NA} contains one symbol for each valid MAC address.

$$V^A_{\mathrm{NA}} \equiv \left\{ n^{(a,A)} : a \in V_{\mathrm{MAC}} \right\} \tag{6.4}$$

Whenever the monitoring node running the timed transducer $\mathcal{M}$ intercepts an NA message for the IPv6 address A, the input symbol converter transforms the message into the symbol $n^{(a,A)}$, the symbol refers to both the IPv6 address A and the MAC address a.

Let the symbol q^*_{MLD} refer to an MLD query for all multicast groups. Let the symbols in the set Q_{MLD} (see Equivalence 6.5) represent all MLD requests; each symbol represents a query for a specific group $G \in V_{\mathrm{IPv6-M}}$. Let the symbols in the set R_{MLD} (see Equivalence 6.6) represent all MLD reports (replies to queries); each symbol $r^{(a,N,G)}_{\mathrm{MLD}}$ contains (1) the MAC address a of the sender, (2) the IPv6 address N of the sender, and (3) the multicast group address G.

$$Q_{\mathrm{MLD}} \equiv \{ q^G_{\mathrm{MLD}} : G \in V_{\mathrm{IPv6-M}} \} \tag{6.5}$$

$$R_{\mathrm{MLD}} \equiv \{ r^{(a,N,G)}_{\mathrm{MLD}} : a \in V_{\mathrm{MAC}}, N \in V_{\mathrm{IPv6}}, G \in V_{\mathrm{IPv6-M}} \} \tag{6.6}$$

Finally, let us define the input alphabet Σ as

$$\Sigma \equiv V^A_{\mathrm{NS-DAD}} \cup V^A_{\mathrm{NA}} \cup \{ q^*_{\mathrm{MLD}} \} \cup Q_{\mathrm{MLD}} \cup R_{\mathrm{MLD}}. \tag{6.7}$$

The output alphabet Λ consists of symbols announcing state changes of the IPv6 address A:

- just begun to be used on an interface with the MAC address a — the set

$$I_{\mathrm{begin}} \equiv \{ \mathrm{Start}^{(a,A)} : a \in V_{\mathrm{MAC}} \}, \tag{6.8}$$

- stopped to be used by the interface with a MAC address a — the set

$$I_{\mathrm{end}} \equiv \{ \mathrm{End}^{(a,A)} : a \in V_{\mathrm{MAC}} \}. \tag{6.9}$$

The output alphabet Λ is defined as follows:

$$\Lambda \equiv I_{\mathrm{begin}} \cup I_{\mathrm{end}}. \tag{6.10}$$

6.4.3 Clocks of the timed transducer $\mathcal{M}$

Timed transducer $\mathcal{M}$ has three clocks,

$$C \equiv \{T^A_{\text{DAD}}, T_{\text{Valid}^{\text{A}}}, T^A_{\text{MLD}}\}, \tag{6.11}$$

where:

- Clock T^A_{DAD} tracks the tentative (or optimistic) phase of the IPv6 address life cycle.
- Clock T^A_{Valid} tracks the validity phase of the IPv6 address A.
- Clock T^A_{MLD} tracks the *Maximum Response Delay* [46] to an MLD query.

Additionally, let us define the following constants:

- t_{RT} is the value of the *RetransTimer* [136].
- t_{Valid} is the value of the valid lifetime contained in RAs [136].
- t_{MLD} is the value of *Maximum Response Delay* in MLD queries. Nevertheless, as the study in Section 6.2 shows, the replies sometimes come late. Therefore, increase t_{MLD} by a small amount of time, for example, 1 second.

6.4.4 Transitions of the timed transducer $\mathcal{M}$

The transitions of timed transducer $\mathcal{M}$ are explained in groups below.

Detection of new address assignments

The initial state S_0 of the timed transducer $\mathcal{M}$ is $(\text{NotUsed}, \epsilon)$. When a host generates the address A on an interface with a MAC address $a \in V_{\text{MAC}}$, the host issues appropriate NS-DAD that the input symbol converter translates to the symbol $s^{(a,A)}$. When the proposed timed transducer $\mathcal{M}$ detects $s^{(a,A)}$, it shifts to the state (InDAD, a) following a rule from

$$\delta_{6.12} \equiv \{((\text{NotUsed}, \epsilon), s^{(a,A)}, \textbf{true}) \to ((\text{InDAD}, a), \epsilon, \{T^A_{\text{DAD}}\}) : a \in V_{\text{MAC}}\}. \tag{6.12}$$

Typically, the address A is not used by another host, the timeout expires, and the timed transducer $\mathcal{M}$ shifts to the $(\text{Address assigned}, a)$ state and $\mathcal{M}$ outputs the newly detected assignment following a rule in

$$\delta_{6.13} \equiv \{((\text{InDAD}, a), \epsilon, t_{\text{RT}} \leq T^A_{\text{DAD}}) \to ((\text{Assigned}, a), \text{Start}^{(a,A)}, \{T^A_{\text{Valid}}\}) : a \in V_{\text{MAC}}\}. \tag{6.13}$$

However, in case the timed transducer $\mathcal{M}$ missed an earlier communication, for example, because it did not receive the messages or it was not yet running, the timed transducer $\mathcal{M}$ receives an NA for the IPv6 address A and registers the other interface identified by the MAC address $b \in V_{\text{MAC}}$ following a rule in

$$\delta_{6.14} \equiv \{((\text{InDAD}, a), n^{(b,A)}, \textbf{true}) \to ((\text{Assigned}, b), \text{Start}^{(b,A)}, \{T^A_{\text{Valid}}\}) : a, b \in V_{\text{MAC}}\}. \tag{6.14}$$

In case that the timed transducer $\mathcal{M}$ detects an MLD report or NA in the $(\text{NotUsed}, \epsilon)$ state sent for an unknown IPv6 address assignment, it learns the assignment following a rule in

$$\begin{aligned}\delta_{6.15} \equiv \{&((\text{NotUsed}, \epsilon), m^a, \mathbf{true}) \to ((\text{Address assigned}, a), \text{Start}^{(a,A)}, \{T^A_{\text{Valid}}\}) : \\ &a \in V_{\text{MAC}}, m^a \in \{n^{(a,A)}\} \cup \{r^{(a,A,G)}_{\text{MLD}} : G \in V_{\text{IPv6-M}}\}\}.\end{aligned} \tag{6.15}$$

The NA detected by a rule in $\delta_{6.15}$ can be, for example, created by Windows Server 2008 R2, Windows 7 and later that join solicited multicast groups after they send NS-DAD, and later send NA with the override flag set. Solaris and recent Mac OS X (for example, 10.7.5) also send unsolicited NA during SLAAC.

MLD traffic monitoring

Once, the timed transducer $\mathcal{M}$ learns that the IPv6 address A was assigned to the interface, it monitors its activity by monitoring MLD queries and reports.

Let us define M^A_{SNMG} to be the IPv6 address of the solicited-node multicast group corresponding to the IPv6 address A [89].

When the timed transducer $\mathcal{M}$ reads a general MLD query (q^*_{MLD}) or an MLD query for the solicited-node multicast group $q^{M^A_{\text{SNMG}}}_{\text{MLD}}$, it shifts to the $(\text{BeingValidated}, a)$ state following a rule in

$$\begin{aligned}\delta_{6.16} \equiv \{&((\text{Assigned}, a), q, \mathbf{true}) \to ((\text{BeingValidated}, a), \epsilon, \{T^A_{\text{MLD}}\}) : \\ &a \in V_{\text{MAC}}, q \in \{q^*_{\text{MLD}}, q^{M^A_{\text{SNMG}}}_{\text{MLD}}\}\}.\end{aligned} \tag{6.16}$$

Several possibilities arise in the $(\text{BeingValidated}, a)$ states:

- If any node replies that the solicited-node multicast group is active, $\mathcal{M}$ shifts back to the $(\text{Assigned}, a)$ following a rule in

$$\begin{aligned}\delta_{6.17} \equiv \{&((\text{BeingValidated}, a), r^{b,B,M^A_{\text{SNMG}}}_{\text{MLD}}, \mathbf{true}) \to ((\text{Assigned}, a), \epsilon, \emptyset) : \\ &a, b \in V_{\text{MAC}}, B \in V_{\text{IPv6}}, B \neq A \vee a = b\}.\end{aligned} \tag{6.17}$$

  Nevertheless, note that another IPv6 address B that is mapped to the same solicited-node multicast group M^A_{SNMG} may confuse the timed transducer $\mathcal{M}$ into believing that the IPv6 address A is still active. Due to the rules for solicited-node multicast group addresses [89], such collisions are not likely.

- If the MLD report comes from an interface identified by different MAC address $b \in V_{\text{MAC}}$ but the same IPv6 address A, the timed transducer $\mathcal{M}$ detects a change in the ownership of the IPv6 address A by a rule in

$$\begin{aligned}\delta_{6.18} \equiv \{&((\text{BeingValidated}, a), r^{b,A,M^A_{\text{SNMG}}}_{\text{MLD}}, \mathbf{true}) \to \\ &((\text{Assigned}, b), \text{End}^{(a,A)}\text{Start}^{(b,A)}, \{T^A_{\text{Valid}}\}) : a, b \in V_{\text{MAC}}, a \neq b\}.\end{aligned} \tag{6.18}$$

- If there is not any reply for the MLD query, the timed transducer $\mathcal{M}$ shifts to the $(\text{NotUsed}, \epsilon)$ state following a rule in

$$\delta_{6.19} \equiv \{((\text{BeingValidated}, a), \epsilon, t_{\text{MLD}} \leq T^{A}_{\text{MLD}}) \rightarrow ((\text{NotUsed}, \epsilon), \text{End}^{(a,A)}, \emptyset) : a \in V_{\text{MAC}}.\} \tag{6.19}$$

- In an unlikely event, during which the timed transducer missed that the IPv6 address was dropped and another host B with the interface identified by a MAC address $b \in V_{\text{MAC}}$ assigned the address A, host B can send an NA message, which is detected by a rule in

$$\delta_{6.20} \equiv \{((\text{BeingValidated}, a), n^{(b,A)}, \textbf{true}) \rightarrow ((\text{Assigned}, b), \text{End}^{(a,A)}\text{Start}^{(b,A)}, \{T^{A}_{\text{Valid}}\}) : a, b \in V_{\text{MAC}}.\} \tag{6.20}$$

Expired address validity

Besides MLD queries and replies, the timed transducer $\mathcal{M}$ detects that a validity of the IPv6 address expired through the clock T^{A}_{Valid} as defined by

$$\delta_{6.21} \equiv \{((\text{Assigned}, a), \epsilon, t_{\text{Valid}} \leq T^{A}_{\text{Valid}}) \rightarrow ((\text{NotUsed}, \epsilon), \text{End}^{(a,A)}, \emptyset) : a \in V_{\text{MAC}}\}. \tag{6.21}$$

Note that the testing in Subsection 6.2.2 revealed that Linux hosts do not perform DAD when they refresh leases. Rules in set $\delta_{6.21}$ do not take refreshed DHCPv6 leases into account. In networks with DHCPv6 and Linux hosts, it is necessary to monitor the leases and reset clock T^{A}_{Valid} whenever a lease for IP address A is refreshed.

Another host performing duplicate address detection

When the timed transducer $\mathcal{M}$ is in one of the $(\text{Assigned}, a)$ states, it can detect another host performing DAD by a rule in

$$\delta_{6.22} \equiv \{((\text{Assigned}, a), s^{(b,A)}, \textbf{true}) \rightarrow ((\text{AnotherDAD}, a, b), \epsilon, \{T^{A}_{\text{DAD}}\}) : a, b \in V_{\text{MAC}}\}. \tag{6.22}$$

Typically, the original host replies that the address A is still active, which is detected by a rule in

$$\delta_{6.23} \equiv \{((\text{AnotherDAD}, a, b), n^{(a,A)}, \textbf{true}) \rightarrow ((\text{Assigned}, a), \epsilon, \emptyset) : a, b \in V_{\text{MAC}}\}. \tag{6.23}$$

However, if the original host is not active anymore, the timeout T_{DAD} expires. Thus, the timed transducer $\mathcal{M}$ learns a new binding and follows a rule in

$$\delta_{6.24} \equiv \{((\text{AnotherDAD}, a, b), \epsilon, t_{\text{RT}} \leq T_{\text{DAD}}) \rightarrow ((\text{Assigned}, b), \text{End}^{(a,A)}\text{Start}^{(b,A))}, \{T^{A}_{\text{Valid}}\}) : a, b \in V_{\text{MAC}}, a \neq b\}. \tag{6.24}$$

In an unlikely event, when the timed transducer misses that the original host dropped the address A and a host with an interface having a MAC address $c \in V_{\text{MAC}}$ assigned the address A, $\mathcal{M}$ learns the assignment by transitions defined by

$$\begin{aligned}\delta_{6.25} \equiv \{((\text{AnotherDAD}, a, b), n^{(c,A)}, \mathbf{true}) \to ((\text{Assigned}, c), \text{End}^{(a,A)}\text{Start}^{(c,A)},\\ \{T^{A}_{\text{Valid}}\}) : a, b, c \in V_{\text{MAC}}, a \neq b, a \neq c, b \neq c\}.\end{aligned} \tag{6.25}$$

Detection of missed address assignments

The timed transducer $\mathcal{M}$ can also detect an MLD report or NA sent by another host in one of the $(\text{Assigned}, a)$ states. Again, this can happen due to a previous message being lost. The rules are defined by

$$\begin{aligned}\delta_{6.26} \equiv \{((\text{Assigned}, a), m^{b}, \mathbf{true}) \to ((\text{Assigned}, b), \text{End}^{(a,A)}\text{Start}^{(b,A)}, \{T_{\text{Valid}}\}) :\\ a, b \in V_{\text{MAC}}, a \neq b, m^{b} \in \{n^{(b,A)}\} \cup \{r^{(b,A,G)}_{\text{MLD}} : G \in V_{\text{IPv6-M}}\}\}.\end{aligned} \tag{6.26}$$

Transition function δ

Finally, transition function δ is defined by

$$\delta = \bigcup_{i \in \{\ 6.12 \dots\ 6.26\}} \delta_i. \tag{6.27}$$

6.5 Evaluation of the IPv6 address assignment tracking

This section describes the experiments with *ndtrack* [93], a tool that follows the proposed approach for IPv6 address assignment tracking. This section is based on text from our previous papers [154, 156], I am the original author of the text.

Firstly, we run three experiments in a laboratory network; the experiments focused on the quality of the monitoring. The final experiment aimed at real network monitoring. In this case, most of the detected devices were not under our control; we could not confirm that *ndtrack* detected all devices. However, we checked that all our devices were detected. All experiments are described below.

6.5.1 A test in a network with MLD Snooping of solicited-node multicast groups

We validated the behaviour of *ndtrack* in a laboratory. We deployed a network with active MLD snooping of the solicited-node multicast groups. Figure 6.9 depicts the network. There was *ndtrack* running on a monitoring node in the network. Additionally, a testing computer was connected to the switch with MLD snooping enabled.

In the first set of experiments, *ndtrack* did not join detected solicited-node multicast groups. In the second set of experiments *ndtrack* joined the multicast groups as described in Section 6.4. Each set of experiments tested several operating systems.

Table 6.2 summarises the results of the experiments. When *ndtrack* did not join the solicited-node multicast groups, the switch did not propagate NS-DADs to the monitoring node. As a result, *ndtrack* did not identify computers that follow the recommended sequence

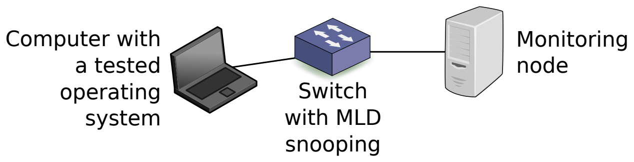


Figure 6.9: The network used for the experiments in a network with active MLD snooping of solicited-node multicast groups.

of messages during the DAD. Windows 8, Mac OS X, and Solaris send additional NAs (as revealed in Section 6.2). The additional NAs allowed *ndtrack* to discover the address assignments even if it did not join the solicited-node multicast groups. When *ndtrack* joined the specified multicast groups during the DAD launched by the tested computer, *ndtrack* successfully detected all operating systems except OpenBSD and static addresses in FreeBSD as expected, see below.

Table 6.2: Effectivity of *ndtrack* in networks with active MLD snooping of solicited-node multicast groups (✓means detected).

	Static addresses		SLAAC addresses	
Monitoring node joined multicast groups	No	Yes	No	Yes
Windows 7 and earlier	-	✓	-	✓
Windows 8	✓	✓	✓	✓
Linux	-	✓	-	✓
Mac OS X	✓	✓	✓	✓
FreeBSD	-	-	-	✓
OpenBSD	-	-	-	-
Solaris	✓	✓	✓	✓

OpenBSD was not detected as expected by Section 6.2. It does not join the solicited-node multicast group corresponding to the tentative address. Consequently, *ndtrack* did not join the solicited-node multicast group. As a result, the switch with activated MLD snooping did not propagate the NS-DADs to the monitoring node.

As already described in Section 6.2, for static addresses, FreeBSD sends NS-DAD before it joins the solicited-node multicast groups corresponding to the tentative address. Therefore, *ndtrack* joined the solicited-node multicast groups corresponding to the tentative address after the tested computer send NS-DAD. As a result, the monitoring node did not discover the address assignment.

6.5.2 Network with stateful DHCPv6

The next experiment tested stateful DHCPv6. Figure 6.10 shows the testing network with computers running Windows 7, 8, 2008 R2, Ubuntu 12.10, and Solaris (one computer for each operating system). As expected, *ndtrack* detected all address assignments. The proposed approach detects DHCPv6 leases.

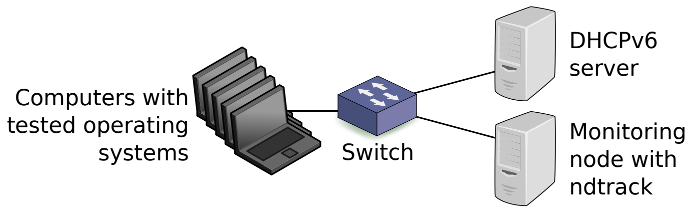

Computers with tested operating systems
Switch
DHCPv6 server
Monitoring node with ndtrack

Figure 6.10: A network utilised for stateful DHCPv6 testing.

6.5.3 Comparison to Other Methods

Figure 6.11 shows the network that we used to compare *ndtrack* with NC polling and *addrwatch* [114]. We tested MLD snooping of solicited-node multicast groups both enabled and disabled. Note that NC polling does not depend on MLD snooping.

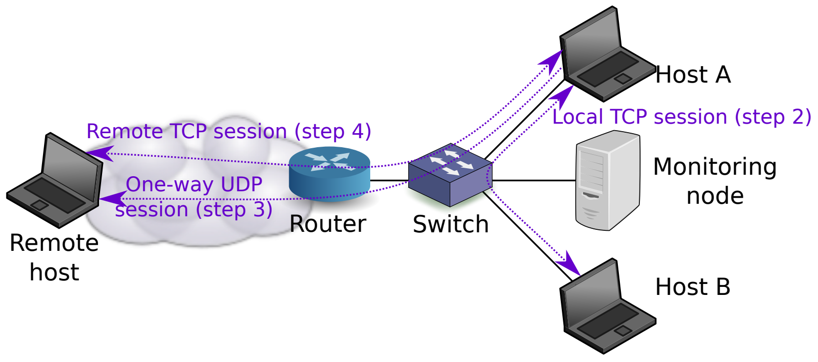

Host A
Remote TCP session (step 4)
Local TCP session (step 2)
One-way UDP session (step 3)
Monitoring node
Router
Switch
Remote host
Host B

Figure 6.11: The test case that compares the *ndtrack* with other methods.

The test case followed several steps for each method. Note that the router also behaved as an MLD querier.

1. Two Linux hosts were deployed in the network.
2. Host A opened a connection to host B. The hosts transferred a file in this connection.
3. Host A initiated a one-way UDP connection outside the network.
4. Host A opened a TCP session to the remote host.
5. Hosts A and B disconnected.

During the experiment, we monitored NC of the router, and the outputs of *addrwatch* and *ndtrack*. Table 6.3 compares the methods.

Data transfers inside a LAN and the outgoing UDP session do not create an NC entry. The router created an NC entry for the monitored host only when the router received a packet destined to the host from the Internet. The entry stayed in the NC after the host was disconnected in the *stale* state.

With MLD snooping disabled, *addrwatch* detected the NS-DADs issued by the hosts when they connected to the network. Additionally, *addrwatch* reported NS messages, issued

Table 6.3: Comparison of our approach with other methods (✓means detected).

	NC polling	addrwatch		*ndtrack*	
MLD snooping	Does not matter	Inactive	Active	Inactive	Active
A, B connected	-	✓	-	✓	✓
Local TCP	-	✓	-	✓	✓
One-way UDP	-	✓	-	✓	✓
Remote TCP	✓	✓	-	✓	✓
A, B disconn.	-	-	-	✓	✓

by the hosts or the router, during the data transfers. However, active MLD snooping did not leak any ND message to the monitoring computer. Consequently, *addrwatch* did not report any activity in the network. Moreover, *addrwatch* did not report that the hosts disconnected from the network even without MLD snooping as no ND message appeared on the network announcing that the IPv6 address was dropped.

Both hosts were successfully identified by *ndtrack* even with MLD snooping active for solicited-node multicast groups. Additionally, *ndtrack* detected that the addresses were no longer used by observing MLD queries and replies.

6.5.4 Real network deployment

The final experiment aimed at long-term monitoring (almost a month) of SLAAC in a network with MLD querying enabled. The network spans two buildings and is available for all employees of the faculty (see Figure 6.12).

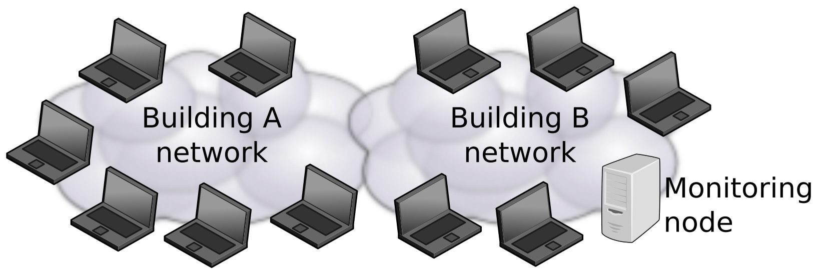


Figure 6.12: The real network monitoring. Most of the computers were not under our control.

We successfully validated that *ndtrack* detected IPv6 addresses of devices under our control among other devices of our colleagues that were active in the network. We also validated that the addresses are correctly identified as no longer assigned after the hosts disconnect or stop using the addresses (see Figure 6.13 for the statistics).

The monitoring node was deployed in building B. To further test the timing of the detection, we connected a device to the network in building A. All addresses of the tested device were correctly identified and later dropped when we disconnected the device.

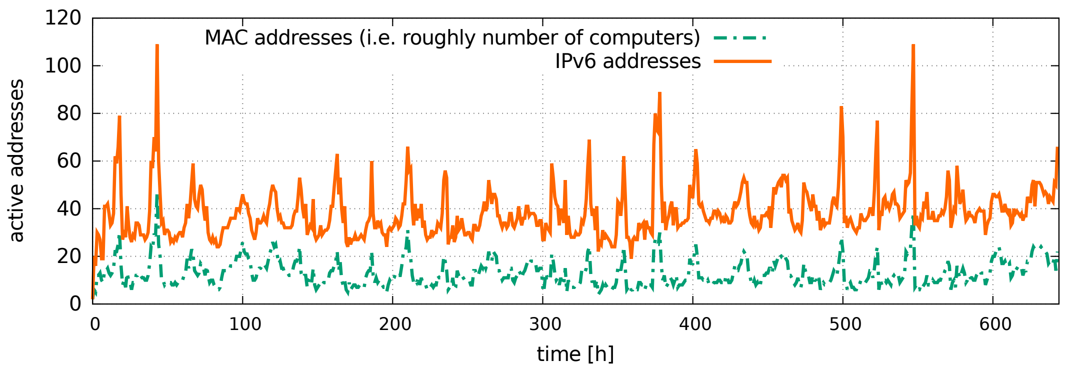


Figure 6.13: Real network monitoring. The number of known addresses rises during working hours and drops at night.

6.6 Considerations about the IPv6 address detection

Timed transducer $\mathcal{M}$ changes state according to the content of ND and MLD messages. Hence, any message not observed by timed transducer $\mathcal{M}$ can desynchronize timed transducer $\mathcal{M}$ and the network. The rules in the set δ contain transitions that try to detect any desynchronization and fix the internal state.

In wireless networks, packet loss is frequent, Yourtchenko and Nortmark measured [197] that a duplicate address is correctly detected in wireless networks only in about 80 % cases. In unjammed wired networks, packet loss is rare, but problems such as network split may hide some ND messages from the observing node [191].

RFC 4541 [30] describes MLD snooping, during which a switch inspects MLD reports and learns the multicast groups that devices on each port listen to. Consequently, the switch can forward multicast traffic only to links with listeners. Most switch vendors implement MLD snooping [191]. Vyncke et al. [191] argue that currently, switches do not have enough memory to track link-local multicast groups. Consequently, switches flood traffic in solicited-node multicast groups to all ports. The proposed IPv6 address assignment tracking based on timed transducers deals even with networks where switches perform MLD snooping for link-local multicast groups since it subscribes to all MLDv2-capable routers multicast group (*ff02::16*) and all solicited-node multicast groups detected by analysing MLD reports in the multicast group for all MLDv2-capable routers. However, in networks spanning a wide area, the delays can cause that a switch in the network receives the MLD report (join) too late and consequently misses the NS-DAD, see Figure 6.14.

6.7 Chapter conclusion

Compared to the related work described in Section 3.3, the proposed IPv6 address assignment tracking based on timed transducer detects all addresses of each node, signals newly assigned addresses immediately, does not poll routers in the network, and distinguishes between states when an IPv6 address was dropped or is merely not used. However, the biggest downside of the proposed IPv6 address assignment tracking is its reliability on the visibility of ND traffic. Each lost message can create inconsistencies between the timed transducer $\mathcal{M}$ state and the network. The proposed IPv6 address assignment tracking works even in

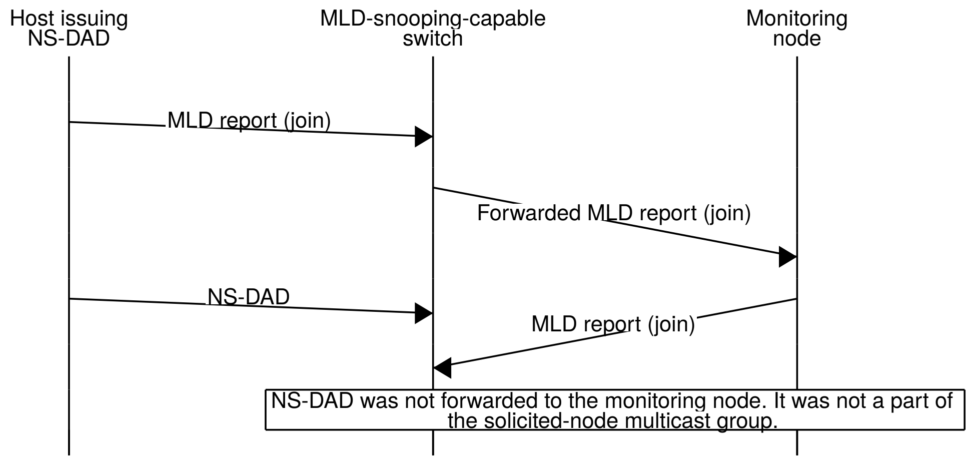

Host issuing NS-DAD
MLD-snooping-capable switch
Monitoring node
MLD report (join)
Forwarded MLD report (join)
NS-DAD
MLD report (join)
NS-DAD was not forwarded to the monitoring node. It was not a part of the solicited-node multicast group.

Figure 6.14: The monitoring node does not see the NS-DAD issued by the host.

networks with MLD snooping active for solicited-node multicast groups (IPv6 multicast is not broadcasted).

With these considerations in mind, the proposed IPv6 address assignment tracking based on timed transducer fulfils Hypothesis 1. If desired the IPv6 address assignment tracking can be further extended in the following way:

- Assignments can be validated with additional NSes. For example, after each new assignment or periodically in a $(\text{Assigned}, m)$ state for $m \in V_{\text{MAC}}$. Nevertheless, the additional messages are visible to end hosts, and consequently, such extension is not suitable for the LI use case.
- In combination with NC polling [81], the monitoring node can reveal assigned IPv6 addresses that were not detected, for example, because some of the traffic was lost and did not reach the monitoring node.

This thesis focuses on instant recognition of address assignments that is transparent to the hosts. Hence, these extensions are not considered in this work.

Chapter 7

Clock-skew-based remote computer identification

Sometimes, it is not possible to deploy partial identity detectors to the LAN of a target of an intercept. Imagine a home network with several individuals or a network of people sharing a flat. In a small network, a deployment of an additional device allowed by an official warrant can raise suspicion of the intercept target. Hence, the local deployment is not possible. The identification has to be done without the address management traffic present only on the LANs. Consequently, the ND-based identification method presented in Chapter 6 is not applicable.

This chapter focuses on remote identification methods that are transparent to identification subjects. The transparency of the identification method is a crucial requirement for the application of the method for LI. This PhD research has studied the clock-skew-based remote computer identification method presented by Kohno et al. [112] as formulated in Hypothesis 2. The method identifies computers based on *manufacturing deviations* [65, Section 2.1] causing small differences in time measurement on each host in the network.

A *fingerprinter* (observer) monitors timestamps of *fingerprintees* (identification subjects). By default, each machine has a constant built-in error in the time measurement. The fingerprinter estimates the error and utilise the estimation as an identifier. TCP timestamps allows transparent clock-skew-based identification by any observer of the TCP flows.

However, during the evaluation of the clock-skew-based identification, we have learnt that the applicability is lower than expected. We [158] revealed that clock value changes, such as those caused by NTP [126], influence clock skew. Later [147], we focused more on the applicability of clock-skew-based measurements in an identification scheme suitable for LI. The final paper [148] formally evaluated the requirements for an accurate clock skew estimation. Additionally, the final paper presented a method that remotely links IPv4 and IPv6 addresses of the same computer. Moreover, the paper [148] elaborated on various use cases of clock-skew-based identification and presented a scenario in which a user can mimic arbitrary clock skew. This chapter is based on the content of the papers on the clock-skew-based identification [147, 148, 158]. I am the author of the text of the papers.

For clock skew measurements, we developed a tool *PC Fingerprinter* (pcf) [153]. The basis of the tool was implemented as a master thesis of Jirásek [104] under my supervision. I rewrote the code into C++ and added support for IPv6, linkage of multiple addresses, and computers with unstable clock skew. Later, bachelor thesis of Franková [69] focused on clock skew measurements and comparison of timestamp sources under my supervision.

Even though this PhD research has revealed that there is only a limited applicability of the method for LI, it improved understanding of several specifics that were not covered by other papers in the area of clock-skew-based computer identification. The contribution of this PhD research in the area of clock-skew-based remote computer identification is:

1. The formal requirements for accurate clock skew measurement (see Section 7.3).
2. The discovery that NTP makes clock skew unstable (see Section 7.4).
3. The method that links the IPv4 and all IPv6 addresses of a single computer (see Section 7.5). (However, Beverly and Berger [16] presented a similar method based on the same ground on a conference held earlier than the paper *Clock-Skew-Based Computer Identification: Traps and Pitfalls* [148] appeared in the journal. Nevertheless, the paper [148] was submitted before the conference.)
4. The guide to mimic a constant clock skew of another computer (see Section 7.6).
5. The study of a short-term measurement in a moderately sized real network (see Section 7.7).

7.1 Hidden identifiers

As introduced in Section 3.2, hidden identifiers are an identifying information that is not obvious. Some hidden identifiers are generally available in network communication, some are present in specific messages, and other hidden identifiers have to be revealed actively. Even though this chapter focuses on clock skew, this section also explains related work in other areas of hidden identifiers. The purpose is to explain why this PhD research has focused on clock skew and not on other hidden identifiers.

Herrmann et al. [88] provide three use cases how hidden identifiers can improve current methods for target-oriented investigations. Their use cases contain fingerprints of encrypted traffic, fingerprints of operating systems, and behaviour analysis. They conclude that fingerprinting techniques can reveal associations between suspects and their actions. Whereas Herrmann et al. did not apply hidden identifiers for LI, and thus the topic of their paper is only loosely related to this thesis, their work applies as an example of other research groups trying to promote the idea of revealing hidden identifiers to improve crime investigation.

7.1.1 Browser identification

A web server can save a small amount of data to a computer in as a cookie. The web browser sends the cookie value in each subsequent request. Originally, cookies were intended to maintain state in otherwise stateless HTTP protocol [137]. Cookies are often used to identify users [5, 137]. Recent studies show that removed cookies are often restored by techniques called evercookie and synchronised between cooperating websites [5, 137].

Nevertheless, web users are identifiable even with disabled cookies. Eckersley set up a web page[1] that collects various attribute values from a browser, for example, the content of HTTP headers including user agent string, screen size, language, time zone, and system fonts. The paper [51] that describes the test reports, that 84 % browsers have a unique fingerprint. When accompanied by information actively probed by Flash and Java, Eckersley uniquely identified 94 % of browsers.

[1]https://panopticlick.eff.org/

Laperdrix et al. [117] have later set up another test page[2] that runs a similar browser recognition test accompanied by additional techniques such as HTML canvas image rendering [128]. Laperdrix et al. observed 17 attributes and reported that Eckersley measurement is not effective as it used to be as Flash is not installed on many devices, especially mobile devices. Laperdrix et al. report that mobile devices are often identifiable by user agent string.

Yinzhi Cao et al. [24] performed the latest test that aimed at identifying specific machines. The goal was to identify the machine even if the user changed a browser. To identify a computer, they instructed the browser to perform several tests including advanced canvas image rendering and audio processing.

Several studies [4, 5, 52, 137] monitored the fingerprinting techniques on the Internet. Results indicate that evercookies, shared cookies, font enumeration, and canvas are commonly used to identify browsers and consequently their users.

Based on the studies conducted by Laperdrix et al. [117] and Eckersley [51], for mobile devices user agent string accompanied by other HTTP headers can serve as an identifier. Nevertheless, the increasing HTTPS traffic limits the possibilities. Many advanced browser identification techniques rely on JavaScript, Flash, and Java. Hence, they are applicable for fingerprinting by a server, but the usability for a passive traffic observer is limited. Additionally, the information in HTTP headers changes, for example, with a new browser version.

In addition, HTTP headers are specific to HTTP protocol only. Their content does not reflect information sent by other protocols such as SIP, SMTP, POP3, IMAP. Hence, a passive identification inspired by browser fingerprinting techniques is limited to HTTP only.

As this thesis aims at a broader identification, we decided not to pursue browser identification techniques and aim at more general techniques.

7.1.2 Behaviour models based on repetitive traffic patterns

Behaviour models aim at passive monitoring of long-term user behaviour. A behaviour model reflects metadata of user behaviour over time, such as the amount of received and transmitted packets and bytes, the size of the packets, required bandwidth, DNS queries. The goal is to exploit the repetitive behaviour of users performing network-related activities, for example, visiting the same set of web servers every day.

Banse et al. [14] linked up to 88.2% of all sessions on a day-to-day basis. However, the successful ratio drops to 60.4% for sessions lasting three hours; for one hour sessions, the successful linking drops lower than 50 %. As a counter-measure, Banse et al. propose frequent changes of IP addresses. The privacy extension addresses of IPv6 [135] change with every reconnection to Wi-Fi networks. Moreover, it is not feasible to wait even for one hour to link the user behaviour with his or her profile. LI has to intercept all data belonging to an intercept target.

Kirchler et al. [110] studied DNS patterns to link user sessions. In their scenario, users change IP addresses of their computers every day. A remote observer tries to identify all user sessions. Kirchler et al. achieved the best ratio of 73 % in identified sessions for highly active users. Only 19 % of active users were tracked completely.

The properties of user behaviour models are not suitable for LI as an IP address has to be identified to belong to a specific user as quickly as possible. As presented in Section 5.2,

[2] https://AmIUnique.org/

IPv6 addresses change once a day or even more frequently. Behaviour models aim at the observation of longer periods and provide the identification with latency. Hence, this thesis does not consider behaviour patterns.

7.1.3 Clock-skew-based identification

Kohno et al. [112] proposed to measure clock skew by monitoring timestamps embedded in TCP segments carrying the timestamp option [18] and ICMP *Timestamp Replies* [161, page 16]. The clock-skew-based identification is based on the idea that the *manufacturing deviations* [65, Section 2.1] in clock crystals cause small differences in time measurement. The differences are unique for each device. The differences cause a small drift of clock with respect to the true time. Each device is manufactured with a specific drift called clock skew. Kohno et al. [112] proposed a method that quickly reveals the clock skew. Due to the uniqueness of the clock skew, the method identifies a computer. As the method was reported to be highly accurate, quick, and applicable to all TCP sessions, we decided to investigate its applicability for LI.

The original idea behind TCP timestamp option [18] is the improvement of TCP performance on large bandwidth delay product paths (paths, where the bandwidth multiplied by the delay is large). The timestamp option has two goals: (1) the protection against wrapped TCP sequence numbers and (2) the measurement of round trip time. The presence of the timestamp option is negotiated during the TCP handshake.

RFC 7323 [18] defines timestamp values as “approximately proportional to real time”. Currently, all TCP sessions initiated by machines running Linux (including Android), Mac OS X, and FreeBSD carries TCP timestamps by default. Kohno et al. observed [112] that servers typically accept a proposal to add timestamp option to all segments.

ICMP sends local timestamp as a reply to *Timestamp Request* probes [161]. Hence, whereas a fingerprinter may gather TCP timestamps passively, ICMP timestamps has to be gathered by active probing.

Clock skew computation

The following description of the clock skew computation is based on my text that appeared in our paper [148].

Let us denote the time reported by clock C at time t (as defined by national standards, that is the *true time*) as $\mathrm{R}_C(t)$. The offset is the difference between two clocks: $\mathrm{off}_{C,D}(t) \equiv \mathrm{R}_C(t) - \mathrm{R}_D(t)$. Assume that $\mathrm{off}_{C,D}$ is a differentiable function in t, then, *clock skew* $\mathrm{s}_{C,D}$ is the first derivative of $\mathrm{off}_{C,D}$. Clock skew is measured in *parts per million* (ppm), meaning that every million time units, the difference between the clocks and the true time increases (or decreases) by the specified amount of units. For example, 20 ppm means that every second, the clock error increases by 20 μs.

Consider C to be the clock of the fingerprinter (observer) and D to be the clock of the fingerprintee (the monitored node) as depicted in Figure 7.1. R_D is not observable by the fingerprinter, instead, it sees packets marked with timestamps delayed by $\epsilon(t)$, $\epsilon(t)$ denotes the delay observed at time t. The delay $\epsilon(t)$ is composed of the processing time at both the fingerprinter and the fingerprintee and the network delay. If ϵ was constant, the first derivative of $\mathrm{off}^{\epsilon}{}_{C,D}(t) \equiv \mathrm{R}_C(t) - \mathrm{R}_D(t - \epsilon(t))$ would have been equal to the first derivative of $\mathrm{off}_{C,D}$. Unfortunately, ϵ is not a constant.

Let us represent observed timestamps from the fingerprintee as *offset points* (x, y) where x is the observation time, either $\mathrm{R}_C(t)$ or the elapsed time since the start of the measure-

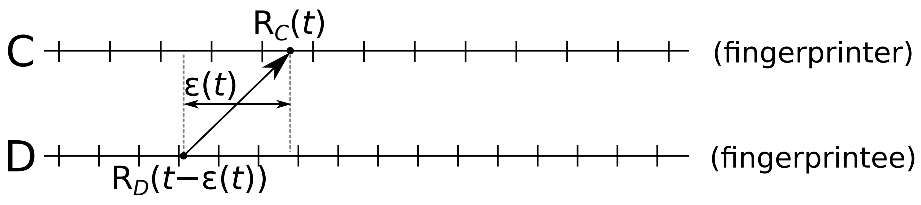


Figure 7.1: Each timestamp of the fingerprintee is delayed by the network and the network stacks of the end hosts.

ment, that is $\mathrm{R}_C(t) - \mathrm{R}_C(t_{\text{start}})$; and y is the observed offset $\mathrm{off}^{\epsilon}{}_{C,D}(t)$. Kohno et al. proposed to estimate clock skew by the slope of the upper bound of all offset points. They have shown that the slope of the upper bound is similar to the slope of the $\mathrm{off}_{C,D}$. Consequently, the first derivatives are similar and the clock skew can be estimated by computing the slope of the upper bound of all offset points. See Figure 7.2.

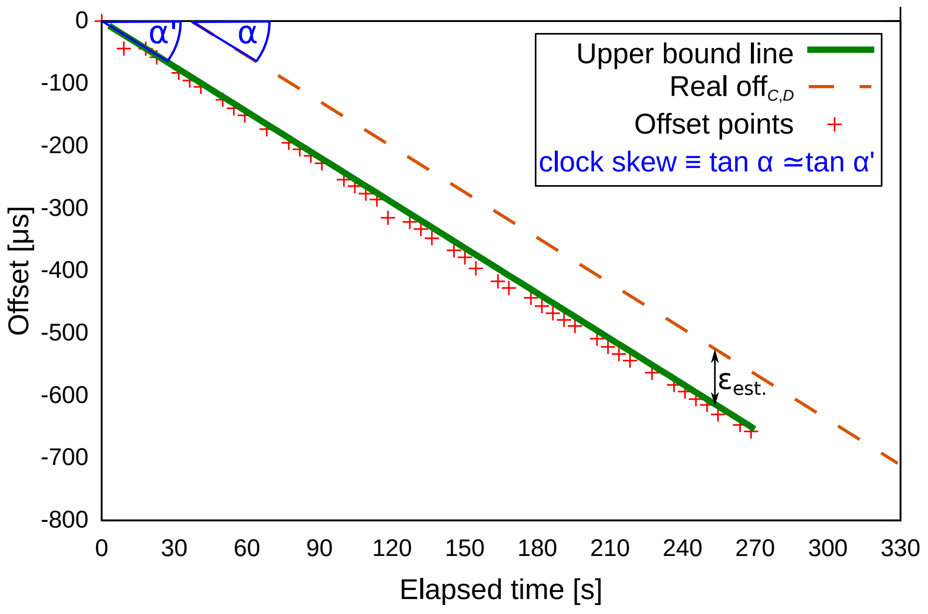


Figure 7.2: Clock skew estimation.

7.2 Related work in clock-skew-based identification

Numerous studies focused on clock-skew-based identification. Each subsection of this section focuses on a specific topic in clock-skew-based identification.

7.2.1 Timestamp sources

The original paper on clock-skew-based fingerprinting by Kohno et al. [112] studied the applicability, advantages and disadvantages of the method. As mentioned above, Kohno et al. used two sources of timestamps: TCP and ICMP. The subsequent research also considers additional sources of timestamps: either carried in protocols on the application layer or the link layer.

TCP timestamps: Unix-related operating systems such as Linux, FreeBSD, Apple Mac OS X, Apple iOS, and Android include TCP timestamp option in the first SYN segment by default. Typically, the server supports TCP options (including Windows servers) and the TCP flow contains timestamps in all TCP segments.

Since the original paper [112], it is known that TCP timestamps are not generated by Windows devices by default. We evaluated that this is valid to this date and Windows 10 still does not support either RFC 7323 [18] or the original RFC 1323 [101] by default. Kohno et al. [112] injected a timestamp as the timestamp option into the first SYN segment. Consequently, Windows clients include TCP timestamps into subsequent TCP segments. A Windows administrator can enable TCP timestamps either in the registry[3] or by the *netsh* command[4].

ICMP timestamps: Cristea and Groza [43] validated that clock skew can be measured from ICMP timestamps on Android. Windows ICMP timestamps have different byte ordering [69] and MAC OS X does not reply to ICMP Timestamp Requests [69].

HTTP timestamps: Murdoch et al. [130, 198] investigated properties of timestamps present in the HTTP protocol. Their goal is to identify the HTTP server. The Date HTTP header [66] contains a timestamp reflecting the time when the web page was generated. Usually, the resolution of the timestamp of the Date header is 1 second resulting in a quantisation error of up to 1 second [198]. Zander et al. [198] compensate the quantisation error by synchronised sampling. During synchronised sampling, the fingerprinter tries to synchronise the probing HTTP requests with the clock ticks of the fingerprintee. Consequently, active fingerprinters can remove the quantisation error even from 1 Hz timestamp sources.

Application layer timestamps: Besides HTTP, many other application layer protocols carry timestamps, for example, XMPP, SMTP, and RTP. Usually, the resolution of these timestamps is in seconds. Hence, to observe the clock skew without the quantisation error, synchronised sampling [198] or similar technique has to be employed.

IEEE 802.11 timestamps: Jana and Kasera [102] and Lanze et al. [116] applied clock-skew-based identification to detection of rogue Wi-Fi access points. The idea lays in fingerprinting another source of timestamps – IEEE 802.11 *beacon frames* (management frames periodically sent by each access point). The fingerprinter is a host that wants to connect to a Wi-Fi network; the goal is to validate a pre-defined clock skew of an access point and detect wireless network spoofing.

Lanze et al. [116] analysed clock skew of 388 access points. They observed that the measured clock skew of the majority of the devices is in the range of -30 ppm to 30 ppm. This observation is in conformance with our study presented in Section 7.7 performed in the network of our university. Additionally, for access points that can run NTP, for example, a Linux-based OpenWRT[5], an attacker can follow the algorithm presented in Section 7.6 and deploy a rogue access point with the clock skew of the original.

[3] http://technet.microsoft.com/en-us/library/cc938205.aspx
[4] http://technet.microsoft.com/en-us/library/cc731258.aspx
[5] http://wiki.openwrt.org/doc/howto/ntp.client

Custom timestamps: An active fingerprinter can force a fingerprintee to provide custom timestamps inserted by an application or a script running on the fingerprintee. For example, Ding-Jie Huang et al. [96] identify HTTP clients as a part of multi-factor authentication by a fingerprinting server. A client that accesses a login web page on the fingerprinting server downloads an AJAX script that periodically sends custom timestamps to the fingerprinting web server. The server leverages these timestamps to compute clock skew of the clients during the connection time.

Summary of timestamp sources Table 7.1 shows the timestamp sources. Passive fingerprinting is transparent to the fingerprintees whereas active fingerprinting generates more traffic into the network which can reveal the fingerprinting to the fingerprintee. Applicability in LI requires passive methods.

Method	Type	Frequency
ICMP	Active	1 kHz
TCP	Passive	10-1000 Hz (OS-dependant)
Application layer protocols	Active or passive	Method/OS-dependant
Application-generated timestamps	Active	Method/OS-dependant
IEEE 802.11 beacon frames	Passive	1 MHz

Table 7.1: Comparison of timestamp sources.

7.2.2 Improvements of clock skew estimates

Several researchers aim to improve the accuracy of clock skew estimates.

Sharma et al. [174] report an error of 0.3 ppm when they combined TCP clock skew estimation with a batch of ICMP packets that compensate for the latency and losses on the network path. This PhD research does not follow this approach for several reasons. (1) The application in LI requires passive detection, see Chapter 4, (2) ICMP timestamps are disabled by default in Apple operating systems, and (3) the successor of ICMP for IPv6, ICMPv6, does not define ICMPv6 Timestamp Request and Reply.

Although Ding-Jie Huang et al. [96] improved the clock skew estimation by linear regression, the reported error is still in the range of ± 1 ppm, that is the same as discussed by Kohno et al. [112]. This PhD research does not employ the linear regression. However, Section 7.3 describes a formula that quantifies the error of the estimation. Following the formula, the inaccuracy of clock skew estimates during long-term measurements may be lower than ± 1 ppm.

Recently, Komang Oka Saputra et al. [139] improved the clock skew detection algorithm by removing outliers with Hough transform. They report an error of 0.59 ppm. Due to the publication date, we did not evaluate their method. A downside of the method is that it does not support a host with unstable clock skew.

7.2.3 Unstable clock skew

Kohno et al. [112] observed changes in clock skew caused by temperature and a power source. According to Kohno [112], clock skew does not change when the user or an NTP [126] daemon adjust clock value. However, other researchers reported changes in observed clock values caused by time manipulation, which invalidates clock skew estimation by computing upper bound of all offset points.

Ding-Jie Huang et al. [96] observed jump points — points over time where the offset changes dramatically (see Algorithm 7.1) but the skew remains the same. Figure 7.3 shows a jump point that occurred after 30 minutes of the clock skew monitoring. The clock skew estimated from data collected during the first 30 minutes of the experiment is similar to the clock skew estimated during the last 30 minutes of the experiment. In both cases, the estimated clock skew is about -353 ppm. However, the upper bound of all offset points estimates clock skew of 133.5 ppm. Obviously, the clock skew estimated from the upper bound is not correct and a fingerprinter has to identify jump points to prevent invalid clock skew estimates.

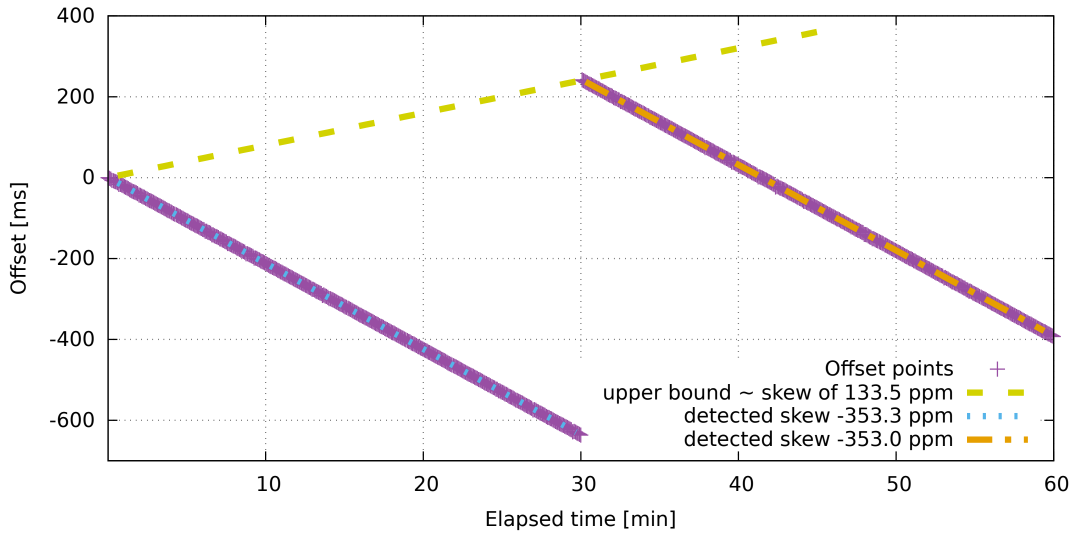


Figure 7.3: A jump point in observed offsets occurred after 30 minutes of a measurement.

Ding-Jie Huang et al. [96] proposed Algorithm 7.1 that detects jump points. When Ding-Jie Huang et al. [96] detect a jump point, they compute clock skew separately before and after the jump point. We also independently observed jump points [104, subsection 5.3.3]. Section 7.4 gives more details on the phenomenon.

Algorithm 7.1. Jump point detection proposed by Ding-Jie Huang et al. [96]:

1. divide timestamps into groups where each timestamp changes less than a threshold compared to the previous timestamp,
2. compute the minimum offset for packets in these groups,
3. compute the difference between minimal offsets of adjacent groups.
4. If all differences are positive or negative, compute median difference and compare each difference to a threshold derived from the median difference. A jump point occurs when the difference between minimal offsets of adjacent groups is greater than the threshold.
5. If groups with a negative difference between minimal offsets are followed with a group g with a positive offset, or vice versa, g is the jump point.

Ding-Jie Huang et al. [96] observed jump points caused by time adjustments and roaming in wireless networks. Section 7.4 provides more details on unstable clock skew caused by NTP and time adjustments in general. Algorithm 7.3 detects changes in observed clock skew including jump points.

According to Cristea and Groza [43], NTP-enabled phones do not run full NTP service. NTP applications in phones do not compensate the inbuilt skew of the oscillator in the device. Instead, the NTP applications periodically shift the device time to the true time, only. Consequently, a fingerprinter sees such synchronisation as a jump point.

7.2.4 IPv4 and IPv6 address linking

Beverly and Berger [16] explored the deployment of IPv6 with the goal of determining if IPv4 and IPv6 addresses listed in DNS for the same service represent the same computer. They call addresses obtained from A and AAAA DNS queries for a single domain name as candidate addresses. Beverly and Berger applied clock skew in an active probing to determine if the candidate addresses belong to the same machine. Beverly and Berger call candidate addresses that belong to the same computer as siblings. To reveal siblings, they propose the Algorithm 7.2.

Algorithm 7.2. Siblings detection proposed by Beverly and Berger [16]:

1. A fingerprinter connects to both candidate addresses resolved for a single domain name. For both, the fingerprinter determines the operating system fingerprint by nmap[6]. If the operating system fingerprints do not match, the candidate addresses are not siblings.

2. The fingeprinter estimates clock skew for both candidate addresses during several TCP connections for each candidate address. Beverly and Berger classify observed clock skew to three groups:

   (a) Timestamp values monotonically increase across TCP connections.

   (b) Timestamp values are influenced by load balancers. Even though timestamp values constantly increase in one TCP connection, timestamp values exhibit a non-linear increase across different TCP connections.

   (c) Timestamp values start from a random value for different TCP connections.

   If the measured category of clock skew for both candidate addresses does not match, they are not siblings.

3. The fingerprinter compares the clock skew value with a threshold to determine true siblings.

As an enhancement, Beverly and Berger [16] classify measured clock skew of negligible value as unknown. Algorithm 7.3 presents the linking algorithm used during this PhD research [148]. In comparison with Beverly and Berger, Algorithm 7.3 is suitable for clients. This PhD research has studied clock-skew-based identification as a sole method. We did not add any additional data such as operating system fingerprinting as LI prohibits active probing.

[6]http://nmap.org/

The idea of comparing clock skew measured for an IPv4 and IPv6 address is common to the research of Beverly and Berger [16] and to this PhD research. Moreover, both papers [16, 148] were published during the same period — our paper [148] was in a review process when Beverly and Berger published their paper [16][7].

Scheitle et al. [171] also studied IPv6 deployment and tried to identify IPv4 and IPv6 sibling addresses based on active fingerprinting. In contrast to Beverly and Berger [16], Scheitle et al. observed many hosts exhibiting non-linear clock skew. Scheitle et al. applied nonlinear splines to determine if candidate addresses are siblings. Although we did not present the exact algorithm, our paper [148] predicted such possibility. We did not pursue the idea presented in the paper because we were interested in short-term measurements. Scheitle et al. [171] present measurements of 9 to 11 hours.

7.3 Accuracy of clock skew measurements

The application of clock-skew-based identification in LI requires that the clock skew of a computer is computed quickly and with high accuracy. The application in multi-factor identification [96] suggests that a quick identification is possible. This section investigates the accuracy both formally and in an empirical study of 24,071 measurements. This section is based on the text from the paper *Clock-Skew-Based Computer Identification: Traps and Pitfalls* [148]. I am the author of the text.

7.3.1 Formal evaluation

The upper bound of the offset points crosses at least two offset points by definition. Let us denote one of these points as an offset point X. Its coordinates are $X = [t_X, \mathrm{off}^{\epsilon}{}_{C,D}(t_X)]$ where t_X is the observation time. In addition, let us denote the amount of time elapsed during period T (of the national standards *true time*) on clock C as $\mathrm{E}_C(T)$. According to Formula 7.1, the y-coordinate of X equals to $\mathrm{off}_{C,D}(t_X) - \mathrm{E}_D(\epsilon(t_X))$.

$$\begin{aligned}\mathrm{off}^{\epsilon}{}_{C,D}(t) &= \mathrm{R}_C(t) - \mathrm{R}_D(t - \epsilon(t)) = \\ &= \mathrm{R}_C(t) - \mathrm{R}_D(t) - \mathrm{E}_D(\epsilon(t)) = \\ &= \mathrm{off}_{C,D}(t) - \mathrm{E}_D(\epsilon(t))\end{aligned} \tag{7.1}$$

Figure 7.4 depicts the geometry of a clock skew estimation. Consider the two offset points K', L' located on the upper bound $\mathrm{b}(t)$. The offset points were observed at time t_1 and t_2. The points K', L', and the point $M' \equiv [t_2, \mathrm{off}_{C,D}(t_1) - E_D(\epsilon(t_1))]$ form a right triangle. In addition, consider the right triangle defined by points $K \equiv [t_1, \mathrm{off}_{C,D}(t_1)], L \equiv [t_2, \mathrm{off}_{C,D}(t_2)$, and $M \equiv [t_2, \mathrm{off}_{C,D}(t_1)]$. Observe that K and L are located on the line representing real offset ($\mathrm{off}_{C,D}(t)$) at time t_1 and t_2. Hence, the points K and L represent the real offset between the clock of the fingerprinter and the fingerprintee at the time t_1 and t_2 that is not biased by the delay ϵ.

From the definition of the tangent function, Formula 7.2 defines the observed clock skew whereas the real clock skew is defined by Formula 7.3.

[7]Moreover, *pcf* estimates clock skew for IPv6 addresses since 2012, see https://github.com/polcak/pcf/commit/c828ea6e39326776704e9bbdfbcefdb1218c9449

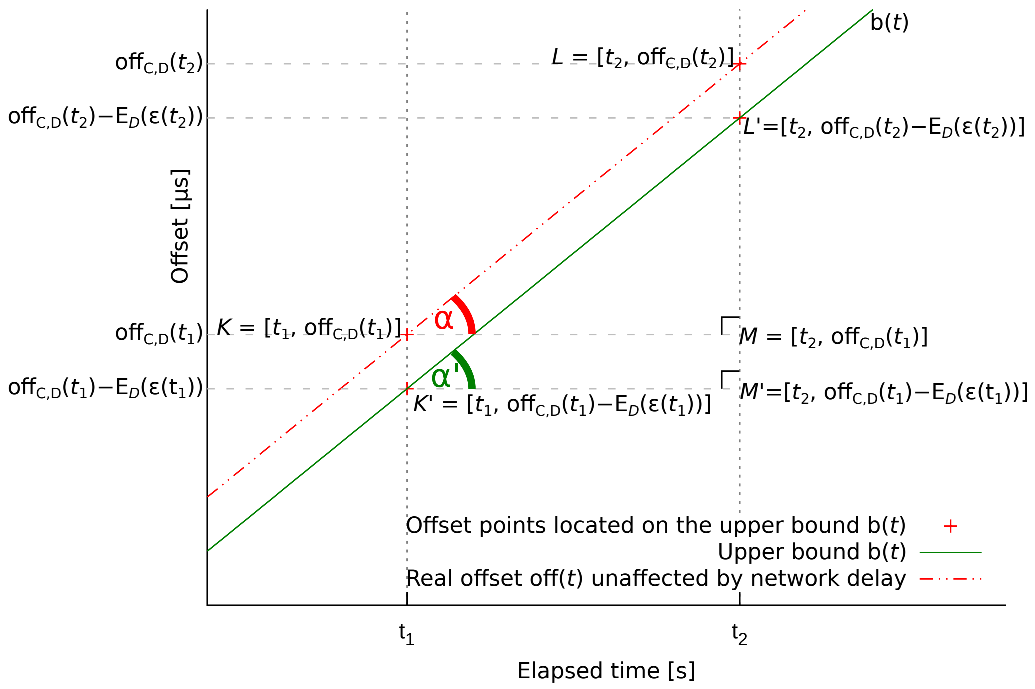


Figure 7.4: Outline of the clock skew computation used for error estimation.

$$\begin{aligned}\mathrm{s}_{\text{observed}} \equiv \tan\alpha' &= \frac{(\mathrm{off}_{C,D}(t_2) - \mathrm{E}_D(\epsilon(t_2))) - (\mathrm{off}_{C,D}(t_1) - \mathrm{E}_D(\epsilon(t_1)))}{t_2 - t_1} \\ &= \frac{\mathrm{off}_{C,D}(t_2) - \mathrm{off}_{C,D}(t_1)}{t_2 - t_1} + \frac{\mathrm{E}_D(\epsilon(t_1)) - \mathrm{E}_D(\epsilon(t_2))}{t_2 - t_1}\end{aligned} \tag{7.2}$$

$$\mathrm{s} \equiv \tan\alpha = \frac{\mathrm{off}_{C,D}(t_2) - \mathrm{off}_{C,D}(t_1)}{t_2 - t_1} \tag{7.3}$$

Combining Formulae 7.2 and 7.3,

$$\mathrm{s}_{\text{observed}} = s + \frac{\mathrm{E}_D(\epsilon(t_1)) - \mathrm{E}_D(\epsilon(t_2))}{t_2 - t_1} \tag{7.4}$$

expresses the dependency of the observed clock skew on the real clock skew and the error introduced by the observed volatile latency ϵ. Observe that Formula 7.4 expresses the observed clock skew as a sum of the real clock skew and the error

$$e \equiv \frac{\mathrm{E}_D(\epsilon(t_1)) - \mathrm{E}_D(\epsilon(t_2))}{t_2 - t_1}. \tag{7.5}$$

Let us denote $\Delta t \equiv t_2 - t_1$ and $\Delta\epsilon \equiv |\,\mathrm{E}_D(\epsilon(t_1)) - \mathrm{E}_D(\epsilon(t_2))|$. Additionally, let us denote the expected accuracy of a clock skew measurement as A. To satisfy the accuracy requirement, the observed error e has to be lower than A as displayed in

$$\frac{\Delta\epsilon}{\Delta t} \leq A. \tag{7.6}$$

Formula 7.6 shows that the quality of the estimation depends on the stability of the network latency $\Delta\epsilon$ and the elapsed time Δt. Note that Δt is weakly lower than the total duration of the measurement as Δt cannot be bigger than the measurement duration. Hence, a longer measurement can yield more accurate estimates as a longer measurement can detect offset points with similar $\Delta\epsilon$ and bigger Δt.

Supposing that the fingerprinter observes only packets going through one network path, the latency introduced by wires and fibres is constant. Papagiannaki et al. studied the single hop delay [140]. Their conclusion is that "there is at least one packet that experiences no queueing in each one minute interval" on a hop (an intermediary device).

Based on the study of Papagiannaki et al., a higher number of packets can yield an upper bound for which the $\Delta\epsilon$ is low. Nevertheless, this effect is only indirect and additional effects have to be considered. For example, the observed delay ϵ depends on the network interconnections as each intermediary device adds to the total delay. In addition, network load or packet size [140] also influence the queueing time. Both longer duration of a measurement Δt and additional packets that can experience lower ϵ relax the conditions on the fluctuation of ϵ as shown in Formula 7.6.

Additionally, the software generating timestamps may increase the variance of ϵ when an already obtained timestamp value is delayed by a data processing algorithm that constructs the network message. For example, a user space code can construct each message based on a different number of kernel calls or there might be a garbage collector running that delays the execution of the program code.

Even more, computer clocks are updated by a constant value. The exact value depends on the underlying frequency of the clock source. This creates quantisation error and it is an important factor in the clock skew computation. Without synchronised sampling [198], it takes longer to compute the clock skew of a source with a frequency of 1 Hz compared to a source with a frequency of 100 Hz, because the time that the clock value does not change contributes to ϵ. Consequently, the increased instability of $\Delta\epsilon$ has to be compensated with bigger Δt to achieve the same accuracy. Nevertheless, an active fingerprinter can synchronise [198] probe packets with clock ticks at the fingerprinter to remove the quantisation error from ϵ.

7.3.2 Empirical evaluation

To examine the relation established by Formula 7.6 between the duration of a fingerprinting Δt and the observed network latency $\Delta\epsilon$ (that influence the number of timestamps needed to compute accurate clock skew estimates), we set up two experimental scenarios during which we monitored timestamps generated by our laboratory computers, all running Red Hat Enterprise Linux 6.6 with Linux kernel 2.6.32.

In the first experimental scenario, we fingerprinted 18 computers on a LAN; there was only one intermediary device between the fingerprinter and each fingerprintee — a switch. We run a Python script[8] repeatedly calling the *time.time()* function. The script sends custom timestamps to the fingerprintee as a payload of HTTP POST requests. Linux kernel automatically added TCP timestamps to TCP headers of the TCP segments carrying the HTTP communication. All computers generated timestamps with a Poisson distribution with expected number of requests between 1 to 10 per second. TCP timestamps had a frequency of 1 kHz whereas the Python-generated timestamps had a frequency of 10 MHz. We included very high frequency Python-generated timestamps to get as accurate timestamps as possible. Since two successive calls of *time.time()* yields different value, we removed the

[8] https://github.com/polcak/pcf/blob/master/exporequests.py

quantisation error. Note that TCP timestamps contained quantisation noise as neither the fingerprinter nor the fingerprintees tried to detect ticks from TCP timestamps.

The first experimental scenario simulated:

1. The possibilities of an active fingerprinter generating custom timestamps (in this case Python-generated) in a user space application, for example, for authentication purposes [96].

2. The possibilities of a passive fingerprinter observing TCP timestamps of an interactive application frequently communicating with a server.

The second experimental scenario aimed at JavaScript-generated timestamps by a script[9] executed by Firefox. The fingerprinter repeatedly monitored four computers on a LAN; each computer was periodically (with a period between 0.1 s and 1 s) sending its current timestamp obtained by the *Date.getTime()* function. The function provides timestamps with 1 kHz frequency. To minimise the quantisation error, we let the JavaScript code to repeatedly obtain a new value from *Date.getTime()* while the returned value did not change. Therefore, the code detected a clock tick and sent the timestamp to the fingerprinter. The second scenario studied the possibilities of an active fingerprinter capable of running JavaScript code in a user browser [96].

In total, we evaluated 9,959 experiments with TCP timestamps, 10,016 experiments with custom Python-generated timestamps, and 4,096 experiments with JavaScript-generated timestamps. During each experiment, we computed current clock skew s_t of the computer based on the actual conditions (for example, temperature). Then, we determined the number of examined timestamps and the elapsed time, after which the clock skew estimation did not leave the $s_t \pm 1$ ppm interval.

The rest of this subsection evaluates the measurements.

Number of timestamps required to get accurate clock skew estimates

Figure 7.5 shows how many timestamps were required to converge to the $s_t \pm 1$ ppm where s_t is the current clock skew of the fingerprintee. For each number of timestamps on the x-axis, the y-axis shows the percentage of experiments that already converged to the expected clock skew range.

According to Figure 7.5, hundreds of packets carrying TCP or Python-generated timestamps were needed in a majority of the experiments, some experiments required thousands of timestamps before the clock skew estimate converged to the ± 1 ppm range of the expected value. We did not observe any special boundary associated with 70 timestamps as reported in previous research [174]. Whereas 41.67 % of experiments with JavaScript-generated timestamps already converged before detecting 70 timestamps, only 0.35 % of TCP experiments and only 0.28 % of experiments with Python-generated timestamps required 70 timestamps or less.

Each HTTP request yielded several TCP timestamps but only one Python-generated or JavaScript-generated timestamp. Hence, a measurement consisting of a particular number of Python-generated or JavaScript-generated timestamps took longer than a measurement computing clock skew from the same number of TCP timestamps. Given a specific number of packets, the share of already converged experiments with Python-generated and

[9] https://github.com/polcak/pcf/blob/master/timestamp46.html

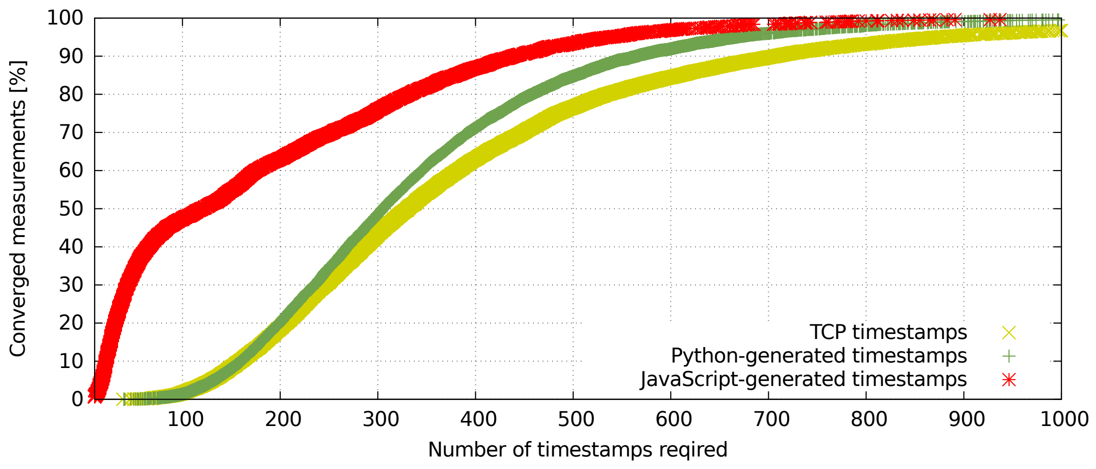


Figure 7.5: The number of timestamps needed to compute accurate clock skew estimates.

JavaScript-generated timestamps were higher compared to the measurements based on TCP timestamps as the TCP timestamps were observed in a shorter time frame.

Time required to get accurate clock skew estimates

Figure 7.6 depicts the percentage of experiments that converged to the $s_t \pm 1$ ppm interval at a specific time after the start of the measurement. Python-generated timestamps took generally much longer to converge. 41.2 % of TCP experiments converged in 20 seconds, 70.6 % converged in 30 seconds and 97.3 % converged in one minute. Only 2.8 % of Python-generated timestamp experiments converged in one minute and 95.1 % converged in 6 minutes. In comparison, 38.1 % of JavaScript-generated timestamp experiments converged in 20 seconds, 53.4 % in one minute, 96.9 % in 3.5 minutes and 99.7 % converged in 6 minutes.

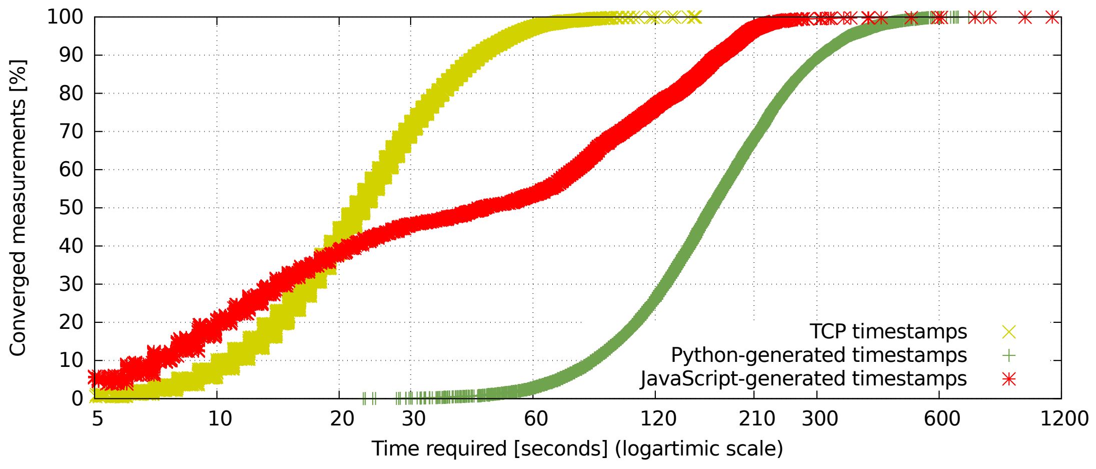
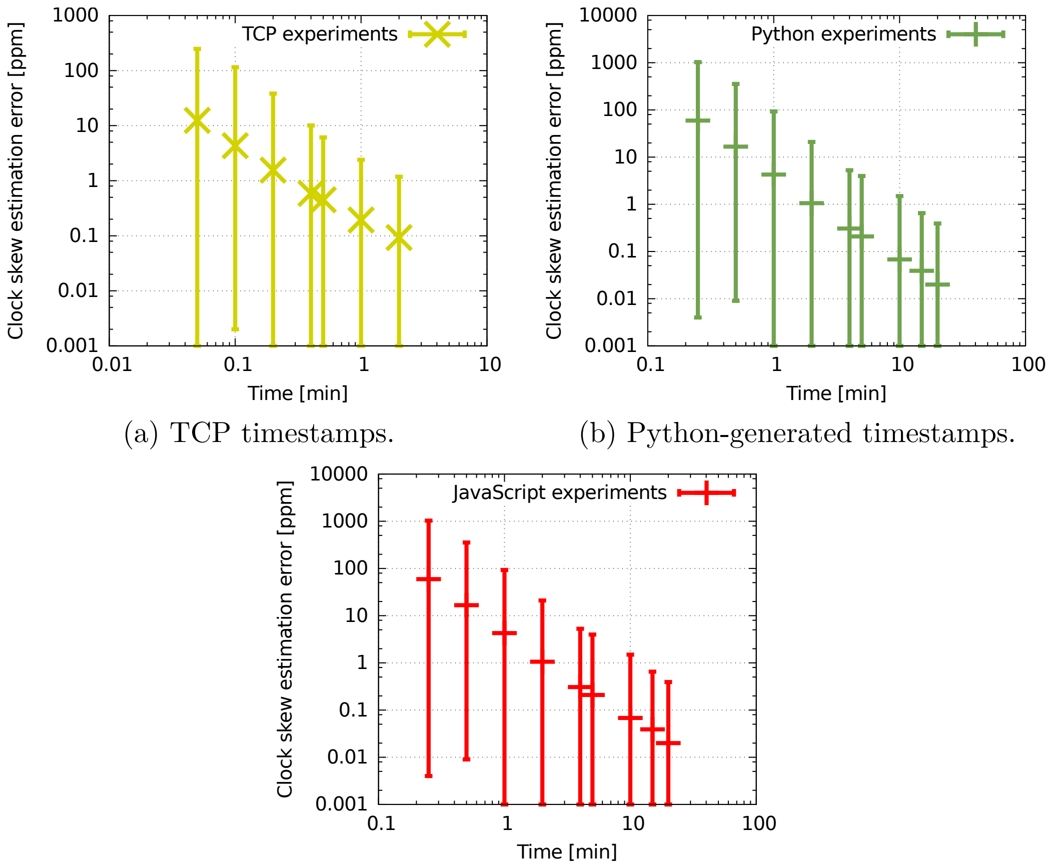


Figure 7.6: Time needed to compute accurate clock skew estimates.

A high volatility of ϵ causes the slow convergence of the Python-generated timestamps. Even though the program received different timestamps after each call of the *time.time()* function, the call of the *time.time()* function requires context switching from the user space to the kernel space and back. Additionally, the Python interpreter can schedule garbage collector or another service routine during the construction of the HTTP request. Finally, the context is switched to the kernel space to send the HTTP request through the network stack. Since the context switching is not deterministic, the variance of ϵ increases and clock skew estimates converge to its correct value longer. JavaScript implementation in the Firefox browser seems to have a lower variance of the ϵ which resulted in a very fast convergence of about 40 % of experiments with the convergence pace better than the pace achievable from TCP timestamps. However, about 60 % of experiments took a longer time to converge in comparison with TCP timestamp fingerprinting.

Observed errors during computation of clock skew estimates

Figure 7.7 depicts the minimal, maximal, and median value of the error in clock skew estimation. The quality of the estimation improves over time, as expected by Formula 7.6.



(a) TCP timestamps. (b) Python-generated timestamps.

(c) JavaScript-generated timestamps.

Figure 7.7: Minimal, maximal, and median value of the error during clock skew estimation.

In summary, the observation confirms the expectations raised by Formula 7.6. The longer an experiment lasts, the less likely it is to get a timestamp that introduces an offset point that incorrectly shifts the upper bound of all offset points. Nevertheless, besides time duration, the variance of delay ϵ also plays a major role.

7.3.3 Conclusion about clock-skew-based identification accuracy

This section proves that the quality of clock skew estimates depends on the duration of the measurement and the stability of the timestamp observation delay. Nevertheless, the number of observed packets influence the quality of the estimate in two ways:

1. The longer the fingerprinting lasts, the more packets are available.
2. The more packets are available, the more probable it is to get timestamps shifted with such similar delay ϵ so that the time difference between the two packets forming the upper bound compensates for the difference in the delay ϵ.

Timestamps sent in the user space seems to be influenced by bigger changes in the delay ϵ. Real-network experiments show that clock estimations based on kernel-generated TCP timestamps have sufficient accuracy in one minute. For user timestamps generated in user space, the desired accuracy is reached after several minutes.

7.4 Influence of time manipulations on clock skew

Experiments in the original paper describing clock-skew-based identification [112] suggest that NTP and other time manipulations do not influence TCP timestamps. However, our experiments with Linux hosts revealed computers with variable clock skew. We observed both sudden changes in timestamp values and little changes in clock skew without visible jumps in timestamp values. Our observations were confirmed by other researchers [96, 171].

NTP [126] is a protocol that synchronises clock between different systems in the network. Typically, a host running an NTP client periodically queries one or several servers and tries to minimise the offset between the local and remote time with regards to observed network latency. The NTP client continuously calls a system call *ntp_adjtime()* that modifies the number of crystal oscillator ticks per second to compensate for the inbuilt error. The goal is to converge to the true time slowly and then maintain the synchronised state. In this mode, NTP does not create sudden changes in system time. As a result, applications running on the system do not observe any sudden changes in system time.

Another use case for NTP is a one-time synchronisation. In this scenario, an NTP client queries an NTP server a replaces the local time with the time on the server with a system call *settimeofday()*. In this case, applications running on the client observe a sudden change in system time. However, *ntpdate*[10] calls *adjtime()* when the difference between remote time and local time is lower than 0.5 seconds.

7.4.1 Time changes via adjtime() and ntp_adjtime() in Linux

First, let us focus on time modifications with *adjtime()* and *ntp_adjtime()*. Both calls modify time in a similar way — the change is gradual. System call *adjtime()* does not maintain any state between calls whereas *ntp_adjtime()* accumulates the correction; hence, it is suitable for continuous time corrections.

Figure 7.8 shows offset points learnt from TCP timestamps of a computer running NTP. Observed clock skew was very close to 0 ppm for the whole duration of the 35-hour-long measurement. The lines in Figure 7.8 shows estimated clock skew of 0.07 ppm for starting 11.7 hours; for the remaining duration of the measurement, the estimated clock skew is

[10] http://www.ntp.org/

-0.04 ppm. In this case, we decided to split the data into two segments, however, the representation can be different, such as splines applied by Scheitle et al. [171].

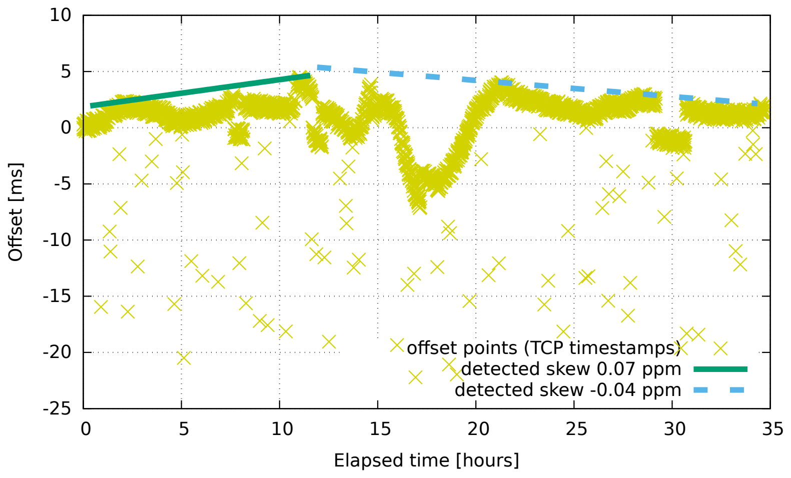


Figure 7.8: Variable clock skew of a computer running NTP.

Figure 7.9 shows a Linux computer with clock skew of about 6 ppm. In this experiment, the computer runs *ntpdate* that calls *adjtime()* every 30 minutes. The clock skew can be computed between the calls of *adjtime()*. In this case, the estimated clock skew is about 6.2 ppm.

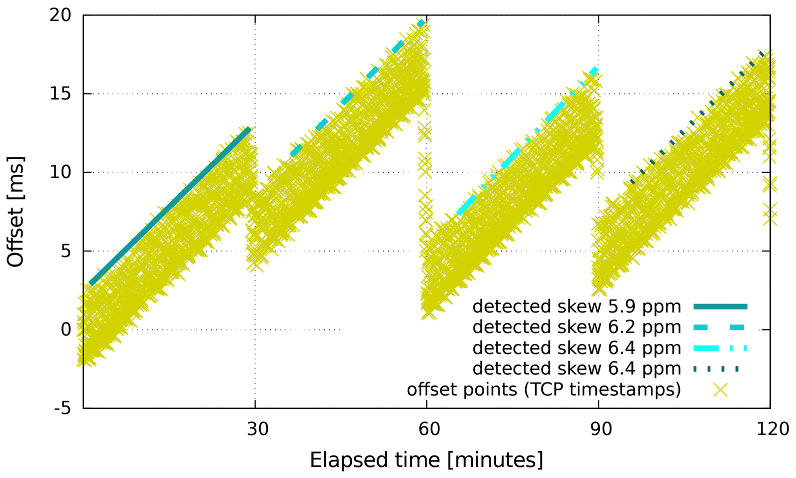


Figure 7.9: Variable clock skew of a Linux computer calling *adjtime()*.

Note that Figure 7.9 shows a short period during which the computer synchronises its clock. By default, *ntpdate* calls *adjtime()* only if the difference between local and NTP time is lower than 0.5 s., which can be overridden with *-B* parameter. Figure 7.10 shows a computer that runs *adjtime()* with *-B* parameter. The observed clock skew changes during the time *adjtime()* slowly corrects the time on the fingerprintee. When the call finishes, the observed clock skew returns to a similar value as before the call of *adjtime()*.

Notice that Figures 7.9 and 7.10 show that calls of *adjtime()* are visible from both TCP timestamps and timestamps inserted by applications in the user space.

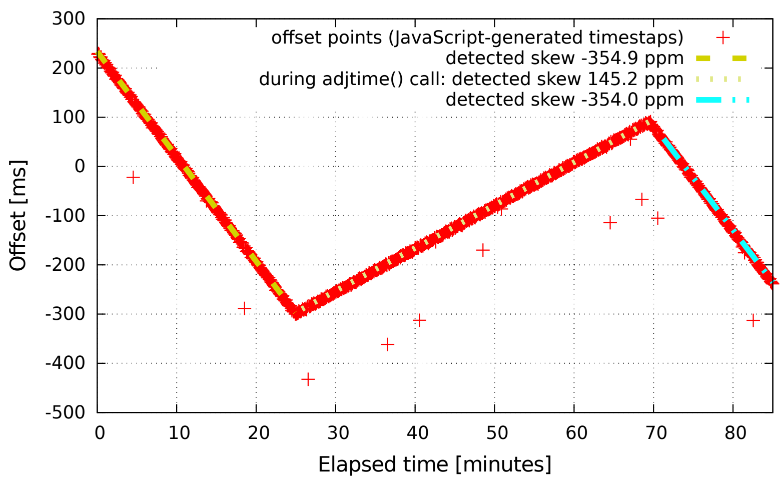


Figure 7.10: Variable clock skew of a Linux computer calling *adjtime()* (long synchronization period).

As the observed fingerprints did not match the expectation of the original clock skew paper [112], we were interested why. We ran experiments with different Linux kernels and revealed that the behaviour changed in 2007 between Linux kernel version 2.6.20 and 2.6.22 [158].

7.4.2 Time changes via settimeofday() in Linux

Another system call that changes local time is *settimeofday()*. This call changes time immediately to the given value. Time modification performed by a user, for example, with *date* POSIX command, typically calls *settimeofday()*.

In Linux, TCP timestamps show the time since the last boot of the operating system (also known as uptime). Consequently, *settimeofday()* does not influence TCP timestamps. In contrast, user space applications requesting the current time via *gettimeofday()* see the time that is modified by *settimeofday()*. As a consequence, timestamps inserted in the user space are influenced by *settimeofday()* while TCP timestamps are not influenced by *settimeofday()*.

Figure 7.11 shows a Linux computer periodically calling *settimeofday()*. Looking at TCP timestamps, the computer has a stable skew of about -354 ppm. Jump points are visible in JavaScript-generated timestamps after each call of *settimeofday()*. Nevertheless, for each segment delimited by subsequent calls of *settimeofday()*, the fingerprinter can detect clock skew of about -354 ppm.

7.4.3 Conclusion on influence of time manipulation on timestamps

Across operating systems, timestamps inserted in user space are obtained by system calls that represent time as observed by the user. The purpose of NTP is to provide time as close to the true time as possible. Hence, every timestamp learnt in the user space carrying the system time is influenced by NTP and other time changes. However, system calls returning time after last boot and similar information are exception.

Our experiments with other operating system showed that time changes are visible in timestamps produced by user space applications in all tested operating systems — Linux,

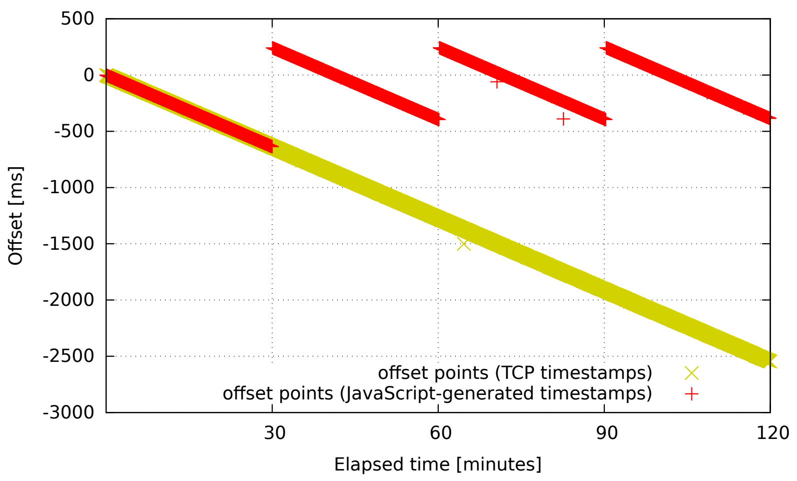


Figure 7.11: Jump points are visible in timestamps obtained from *gettimeofday()* for each *settimeofday()* call. TCP timestamp values do not change after a call of *settimeofday()*.

Windows, OpenBSD, Apple iOS, and Apple Mac OS X. At TCP level, NTP does not change clock skew of a Windows computer. Both Linux and OpenBSD propagate NTP changes to TCP timestamps. Apple operating systems exhibit highly unstable clock skew on TCP level. We were not successful [147] in fingerprinting Apple devices from TCP timestamps.

Table 7.2 summarises the influence of time changes on observed clock skew. Figures 7.2, 7.4, and 7.11 (TCP timestamps) are examples of stable clock skew. Computers with stable clock skew can be linked over time because the clock skew does not change even if the computer changes its location or IP address. Figures 7.9, 7.10, and 7.11 (JavaScript-generated timestamps) depict observable changes caused by clock manipulation. A fingerprinter can detect jump points, estimate clock skew between jump points, and consequently learn the clock skew of the computer. Then the fingerprinter can track the computer over time in different locations and with different IP addresses, similarly to stable clock skew. Offset points of computers with unstable clock skew do not form a line. Figure 7.8 depicts an example of unstable clock skew close to 0 ppm. Computers with clock skew close to 0 are indistinguishable between each other over time. Figure 7.12 depicts an example of offset points computed from TCP timestamps produced by Apple operating systems.

Table 7.2: Influence of time manipulations on clock skew fingerprinting.

	TCP			Userspace timestamps
	Continuous NTP	*adjtime()*	*settimeofday()*	
Linux	Unstable, 0 ppm	Observable	Stable	Observable changes
OpenBSD	Unstable, 0 ppm	Observable	Stable	Observable changes
Windows	Stable			Observable changes
Apple	Highly unstable, large			Observable changes

7.5 Applicability for IPv6 addresses

Network layer does not modify the payload produced by upper layers. Consequently, both TCP timestamps and timestamps inserted by applications are not influenced by network layer. Hence, this PhD research has pursued the idea that IPv4 and IPv6 addresses are linkable via clock skew [69, 148]. IPv4 and IPv6 address linking based on active fingerprinting was independently published by Beverly and Berger [16], see Subsection 7.2.4 for more details.

This PhD research does not consider clock skew to be stable over time. Instead, *pcf*, the fingerprinting tool developed as a part of this PhD research, continuously computes clock skew for each active IP address in the network based on the Algorithm 7.3. Appendix A shows an example of an application of Algorithm 7.3.

Algorithm 7.3. Proposed clock skew estimation

1. Initialize *previous batch* to empty.
2. Collect a batch of timestamps. The batch can be determined by duration or the count of timestamps.
3. For each new batch, estimate the clock skew based on offset points generated for the timestamps in the *current batch*. If the *previous batch* is empty, set *current batch* as *previous batch* and go to step 2.
4. Compare the estimate for *current batch* to the previous estimate. If the estimate differs by more than 10 ppm (by default), clock skew changed or a jump point was detected, go to step 6. Otherwise, go to step 5.
5. Append offset points in *current batch* to the *previous batch* and estimate clock skew based on all merged offset points. Go to step 2.
6. Ignore *current batch* as the observed clock skew changed or the batch contains a jump point. Go to step 1.

For each batch created by Algorithm 7.3, *pcf* creates a triplet consisting of:

- the clock skew estimate,
- initial time from which the clock skew estimate is valid (observed time of the first offset point in the batch),
- final time until which the clock skew estimate is valid (observed time of the last offset point in the batch).

Hence, for stable clock skew, *pcf* detects a single triplet; for unstable clock skew, *pcf* detects multiple triplets, each valid for a particular period. For stable clock skew, *pcf* continuously improves the clock skew estimate (see Formula 7.6) and prolongs its validity. For unstable clock skew, *pcf* detects jump points and periods where clock skew changed.

This PhD research considers two sequences of clock skew triplets (each observed for an IP address) to be linkable, meaning both IP addresses belongs to the same anonymity set, which (hopefully) represents a single computer, if one of the following requirements holds [148]:

1. Both clock skew estimates are stable (both sequences have a single triplet) and both estimates are within the range of ± 1 ppm. In this case, it does not make a difference if both addresses are active during the same period or not.

2. Both clock skew estimates were within the range of ±1 ppm during the periods when both addresses were active. In this case, there has to be at least one period during which both addresses were active; when the clock skew of one of the addresses changed, the other clock skew changed soon to a similar value, for example, a jump point was detected for both IP addresses.

The first requirement is the same as originally used by Kohno et al. [112]. The latter requirement deals with changes in clock skew as Section 7.4 presents.

Additionally, *pcf* detects addresses with a long duration of inactivity, by default one hour. When an IP address is probed after a long inactivity period, the clock skew estimation starts from the beginning. The restarted clock skew estimation allows *pcf* to deal with address reassignments to different computers.

We evaluated the linkability of IPv4 and IPv6 addresses in a laboratory network [148]. For the evaluation, we used the same data as in the empirical study described in Subsection 7.3.2 — the data from clock skew estimations based on TCP timestamps, custom Python-generated timestamps, and custom JavaScript-generated timestamps. All experiments were conducted using both IPv4 and IPv6. All clock skews were stable in this evaluation. For all computers, we checked that clock skew estimates for both IPv4 and IPv6 addresses converged to the same value.

In addition, we evaluated an Apple Mac Mini, an Apple Mac Book (Mac OS X 10.7.5 and earlier versions), an Apple iPad (iOS 9.3.5 and earlier versions), Windows 7, Windows 8.1, and FreeBSD computers. For these operating system, we tested JavaScript-generated timestamps and TCP timestamps.

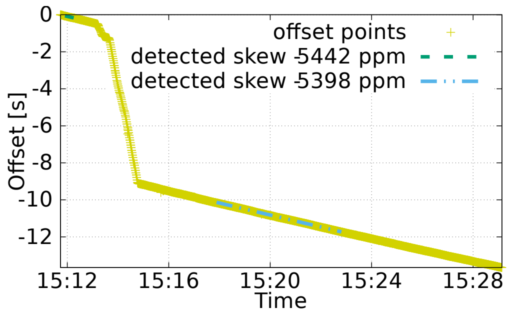
Firstly, we tested the computers without time modifications. The results are following:

- JavaScript-generated timestamps carried over IPv4 and IPv6 converged to the same values.
- TCP timestamps converged to the same values as JavaScript-generated timestamps for all operating systems except Apple.
- TCP timestamps of Apple computers exhibited very large clock skew. Generally, the computed values for IPv4 and IPv6 did not match. However, as Figure 7.12 shows, there are observable changes in both IPv4 and IPv6 traces generated by an Apple computer. Note that in this case, there were no time changes on the computer. Although the clock skew estimations for IPv4 and IPv6 differ by hundreds of ppm, the shapes formed by the offset points are similar. We believe that such computers can be linked with an approach similar to nonlinear splines developed by Scheitle et al. [171].

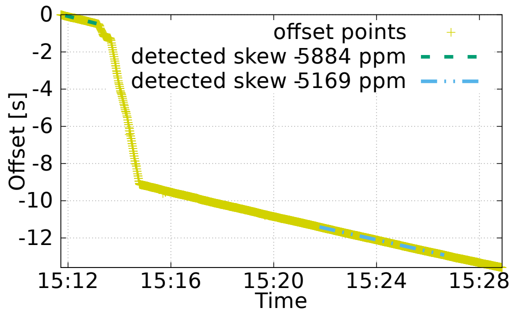
7.6 Guide to mimic clock skew of a different computer

A possibility to influence the observable clock skew of a computer is a major downside for the identification of the intercept target during LI. Hence, this section focuses on the possibility to mimic an arbitrary stable clock skew [148].

Also, note that Ding-Jie Huang et al. [96] applied clock-skew-based identification in a multi-factor authentication. A server estimates a clock skew during the time a client authenticates and if the clock skew estimate matches one of the values previously seen for



a) IPv4 address.



b) IPv6 address.

Figure 7.12: Clock skew measurement of an Apple device based on TCP timestamps.

the user account, the server accepts the authentication. In a case of a previously unseen clock skew value, the server challenges the user with another authentication method, such as SMS, or performs another action, for example, the server sends an alert message. An adversary with stolen credentials can mimic clock skew of a victim computer to pass the clock-skew-based authentication.

An NTP daemon maintains a file called *driftfile* where it stores current correction for the built-in clock error. The goal is to maintain accurate time even after the NTP daemon is restarted. After the restart, the daemon sets the correction stored in the *driftfile*.

To mimic clock skew of another computer, an adversary can follow the Algorithm 7.4.

Algorithm 7.4. Mimicking clock skew of a different computer [148]:

1. Run an NTP daemon and find the clock skew of the attacking computer, for example, from the *driftfile*.

2. Find the clock skew of the victim, preferably by fingerprinting the victim computer by the attacking computer (still running the NTP daemon).

3. Add the clock skew learnt in step 2 to the value stored in the *driftfile* in step 1. Use the sum as the correction of attacking computer clock, for example, by restarting the NTP daemon on the attacking computer and immediately stopping the daemon.

By applying Algorithm 7.4, we were able to reproduce clock skew of a different computer in our laboratory [148].

A user that tries to evade clock-skew-based detection can randomise clock skew of his or her computer during boot or as a part of network access procedure following Algorithm 7.5.

Algorithm 7.5. Randomising stable clock skew of a computer:

1. Generate a random clock skew value, preferably in the range of observed clock skews in real network reported in Section 7.7 or by Lanze et al. [116].

2. Apply the generated random clock skew, for example, by starting and immediately stopping the NTP daemon.

A fingerprinter of a user with randomised clock skew by Algorithm 7.5 estimates different clock skew after each execution of the Algorithm 7.5. Between the changes, the fingerprintee exhibits stable clock skew.

7.7 Real world measurements

The final test of clock-skew-based identification aims at real world deployment. Originally, we performed the experiment for IPv4 [147] and later, we expanded the test for IPv6 [148]. This section is based on these experiments and the text of both papers [147, 148]. I am the author of the text in both papers.

We passively fingerprinted devices in our faculty network based on TCP timestamps. The goal was to observe the network traffic in the same manner as an LI system that tries to identify the computers.

During the testing, we did not focus on specific devices. For privacy reasons, we did not compare the gathered information to external sources. However, based on the fingerprinting location, we fingerprinted following devices:

- laboratory computers (Windows, Linux, and FreeBSD),
- desktops and laptops of the faculty staff with timestamps enabled (Linux, FreeBSD, Apple Mac OS and iOS, and possibly Windows with manually enabled TCP timestamps),
- mobile devices connected to the faculty Wi-Fi (for example, Android, iOS),
- faculty servers (mostly Linux and FreeBSD),
- remote servers accessed by the computers mentioned above,
- remote clients that connected to local servers with TCP timestamps enabled.

We performed a continuous measurement during several days. During working hours, we observed about 350–649 active IPv4 addresses and 100–196 active IPv6 addresses. During nights and weekends, we observed about 120–170 active IPv4 addresses and 15–40 active IPv6 addresses. We did not observe any substantial changes in observed clock skews. The following text focuses on data obtained in the afternoon of a working day.

At that moment, the fingerprinter estimated clock skew for 646 active IPv4 addresses and 177 active IPv6 addresses. Figure 7.13 shows the fingerprinted distribution of clock skew estimations for IPv4 and IPv6 addresses. The majority of observed clock skew is close to zero. Hence, the devices close to 0 ppm are indistinguishable by the clock skew value.

Note that for 92 IPv4 addresses and 30 IPv6 addresses, the estimated clock skew was lower than -1000 ppm or higher than 1000 ppm. Such addresses exhibited similar behaviour as Apple devices in our laboratory, for example, the clock skew was not stable similarly to Figure 7.12. We do not consider addresses with clock skew below -1000 ppm and above 1000 ppm as the laboratory measurements [147] show that the observed values from TCP timestamps of Apple operating systems do not match clock skew estimation from JavaScript-generated timestamps and visual observations.

Figure 7.13 shows only 20 active IPv4 addresses and 5 active IPv6 addresses with clock skew estimated between -1000 ppm and -100 ppm. Additionally, the are 15 active IPv4 addresses and 2 active IPv6 addresses with clock skew estimated in the range between 100 ppm and 1000 ppm. Clock skew estimation lower than -52.3 ppm and higher than 85.7 ppm are unique in our dataset; that is the computers are identifiable by clock skew. Some of these addresses exhibited long-term stable clock skew, for example, one IPv4 address with estimated clock skew of 501.1 ppm was fingerprinted for 105 minutes.

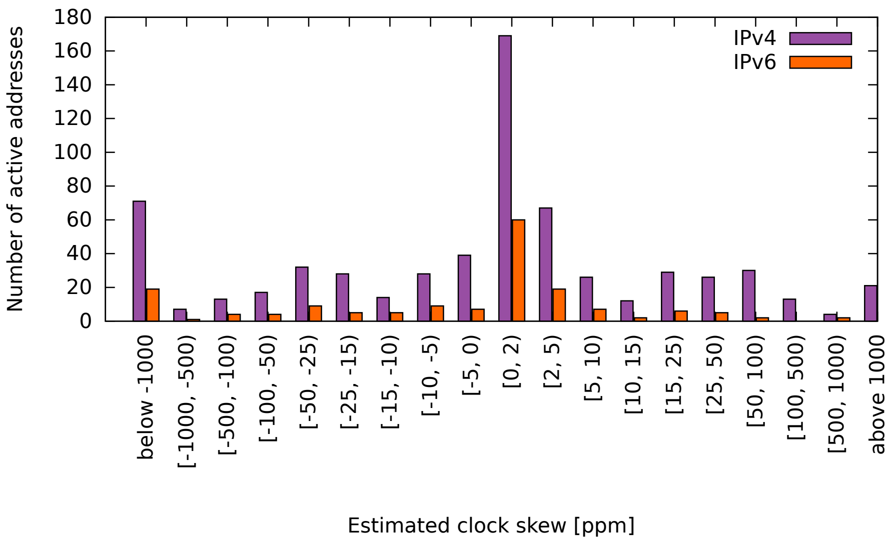


Figure 7.13: Histogram of clock skew distribution in real network. Note that the range of bins is not uniform.

The estimated clock skew for 80 % IPv4 addresses and 79 % IPv6 addresses were in the range of -100 ppm to 100 ppm. Let us focus on these addresses.

Figure 7.14 shows the distribution of clock skew estimates for the range of -100 ppm to 100 ppm; for each clock skew estimate (x-axis), it reports the number of other estimates in the anonymity set defined by the ±1 ppm range, that is possibly of the same computer. The largest anonymity set for IPv4 contains 223 IPv4 addresses; the largest anonymity set for IPv6 contains 72 IPv6 addresses. Both largest anonymity sets are close to 0 ppm. Hence, computers having assigned 34.5 % of all IPv4 addresses and 40.7 % of all IPv6 addresses are indistinguishable. Most probably, these addresses were assigned to computers running NTP.

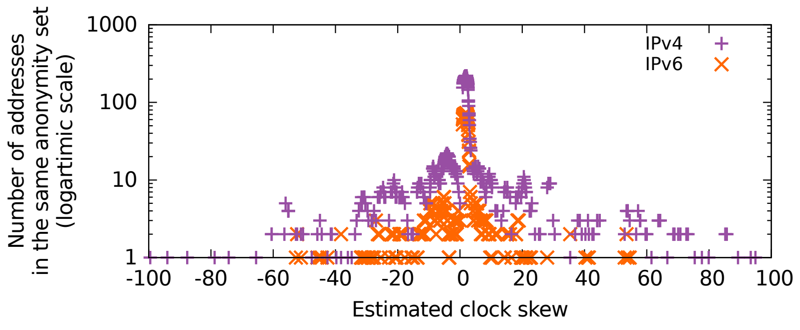


Figure 7.14: The estimated clock skew and number of addresses with similar clock skew in the range of -100 ppm to 100 ppm.

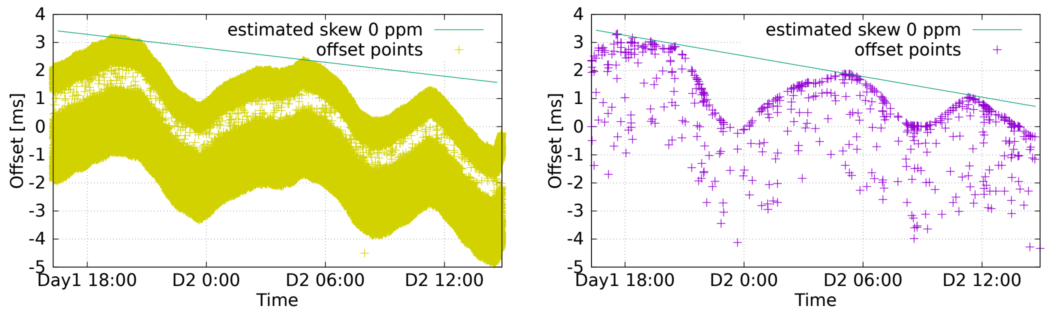
Even without hosts running NTP continuously, the majority of devices appear in the same anonymity set with multiple other devices. Between -100 ppm and 100 ppm, only 21 IPv4 addresses were not in an anonymity set with another IPv4 address. Quick, unique identification of computers in a moderately sized network is not possible with restrictions compatible with LI. These observations confirm observations of Lanze et al.[116].

The manufacturers aim at producing clocks that are close to 0 ppm. Hence, it is not surprising that in a moderately sized network, a majority of computers exhibits clock skew reasonably close to 0 ppm. Whereas the laboratory experiments uniquely distinguished 20 computers, the real network deployment revealed the disadvantages of clock-skew-based identification.

7.8 Applications of clock-skew-based identification

Although the real deployment of clock-skew-based identification for LI was not successful, other applications for clocks-skew-based identification exist. This section focuses on the applicability of clock-skew-based identification.

A fingerprinter can aim at short-term fingerprinting, for example, during multi-factor authentication [96] or identification for LI. Nevertheless, long-term fingerprinting can reveal additional information, such as geographical position [130], and link addresses based on clock skew changes [171]. Figure 7.15 shows an example of a computer from a real network measurement. The computer was running NTP. The amount of IPv4 traffic was much larger compared to IPv6 traffic. However, similar changes in clock skew caused by continuous NTP synchronisation are visible. A long-term fingerprinter can link these addresses whereas short-term fingerprinter does not see the changes in a short timeframe and estimates clock skew close to 0 ppm. See Figure 7.12 as another example of a long-term measurement that enables IPv4 and IPv6 address linking.



a) IPv4 address. b) IPv6 address.

Figure 7.15: Real-network observation of a Linux computer running NTP.

Based on the fingerprinting duration and active or passive observations of timestamps (see Table 7.1), it is possible to classify fingerprinters into four categories [148].

1. Passive short-term fingerprinting aims on quick identification of a fingerprintee with the applications in LI or rogue access points identification [116]. Both this PhD research and the research of Lanze et al. [116] show a limited applicability in small networks only. Fingerprintees can spoil the fingerprinting by clock skew modifications, such as the clock skew randomization achieved by Algorithm 7.5.

2. A passive long-term fingerprinter can observe changes in clock skew caused by NTP or temperature changes. Consequently, the fingerprinter can link all IP addresses used by a specific computer, learn geographical location [130] or disclose computers behind network address translator [112].

3. An active short-term fingerprinter can use clock-skew-based identification during multi-factor authentication [96] or improve the estimates [174] compared with short-term passive fingerprinter. However, the disadvantages connected to the limited uniqueness of clock skew and possible influence of the fingerprintee on its clock skew also apply to active short-term fingerprinting.

4. An active long-term fingerprinter can improve the quality of the clock skew estimates [174] compared to passive-long-term fingerprinter. Active long-term fingerprinting can be applied in deanonymization [198], virtual PC and honeypot detection [112], and IPv4 and IPv6 linkability [171].

7.9 Chapter conclusion

This chapter describes remote computer identification based on clock skew estimation. Previous research indicates that unique short-term identification is possible [96, 112, 174] and that passive clock skew evaluation is possible [112]. However, our research revealed that the main source of timestamps for passive fingerprinting, TCP, has became influenced by NTP changes. This chapter identifies that this change impacts mostly short-term fingerprinting. The results also suggests that long-term clock-skew-based-fingerprinting is a viable choice for deanonymization [198], virtual PC and honeypot detection [112], and IPv4 and IPv6 linkability [171].

The study of real network clock skew distribution, presented in Section 7.7, shows that most of the real devices exhibit clock skew in the range of -100 ppm to 100 ppm with the majority of devices close to 0 ppm. Consequently, a short-term fingerprinter cannot reliably identify computers solely by clock skew.

Clock-skew-based identification does not work in conditions restricted by Hypothesis 2. Remote short-term passive fingerprinting based on TCP timestamps is not reliable. However, clock-skew-based identification can be employed in the following use cases:

1. A long-term passive fingerprinter can link addresses of NTP-enabled hosts by observing the shape of the upper bound, see Figures 7.12 and 7.15 for examples.

2. An active fingerprinter can combine clock-skew-based fingerprinting with another identification method, such as browser fingerprinting. As browser fingerprinting alone can uniquely identify [24, 117] a computer or a browser running on a computer, browser fingerprinting and clock-skew-based fingerprinting combined provides additional possibilities for unique identification.

Nevertheless, both modification does not allow clock-skew-based identification to be applicable in LI for a moderately sized network. Short-term fingerprinting is applicable in small networks, especially in a controlled environment where users cannot manipulate time.

Law enforcement can benefit from clock-skew-based identification during a forensic investigation. For example, *pcf* [153] allows an investigator to estimate clock skew from offline traces stored in a pcap file. With expert knowledge, the investigator can reveal clues for his or her investigation, such as the linkability of several addresses belonging to the same computer, see Figures 7.12 and 7.15 for examples.

Chapter 8

Identity graphs

The ability to link identifiers is crucial to identify all data of the intercept targets. For example, the interception of CC is often based on IP addresses because IP addresses appear in the header of every IP datagram [90, page 272]. Nevertheless, the IP addresses are assigned dynamically. Hence, an LI system has to link a stable identifier such as a RADIUS username, a network access identifier, a cable modem identifier, or a stable MAC address to the dynamically assigned IP address [90, page 272].

Recently, the significance of application-layer-identifier-based LI grows [7, 90, 187, 195]. Typically, such intercept targets an application layer identifier such as an e-mail address [61] or a VoIP identifier [63]. Consequently, the interception of CC is based on a dynamic IP address or a dynamic flow identifier that are contained in header fields of all packets. An LI system has to link the application layer identifier to the dynamic identifier that appears in all packets of the session.

This chapter focuses on Hypothesis 3 by searching for a model based on formal rules for cross-layer linking of partial identities discovered by different partial identity detectors. The goal is to link an input identifier (identifying the target of the interception) to other identifiers allowed by a warrant.

The input identifier is a long-term identifier identifying a user, a household, a computer, or a computer network interface. Depending on the wording of the warrant (see Section 5.5 for the problem statement), some linking might be allowed or forbidden. For example, a warrant for intercepting traffic of an IP address forbids linking the IP address to another IP address, including all IP addresses of the same network interface. Another warrant for interception of traffic of an interface (or a computer) identified by an IP address allows linking the IP address to other addresses of the interface (or the computer).

To incorporate identification methods that are not completely accurate (for example, clock skew, browser fingerprinting techniques), the model should include accuracy, which disqualifies linkage through multiple inaccurate observations.

For digital forensic, the model should be aware of time so that the dynamic identifiers can be tracked over time and the model should support time-related reasoning on the gathered knowledge.

This chapter defines identity graphs that allow cross-layer linkage of information from various partial identity detectors in conformance with Hypothesis 3. The proposed model is based on a graph representation of identity information; vertices represent identifiers and edges reflect the linkage between two identifiers. Identity graphs support operations that link identifiers following constraints based on the wording of a warrant and other parameters, such as the inaccuracy of the linkage. Appendix B provides examples of queries in identity graphs. The definition of identity graphs and the operations is extensible. Hence, this chapter provides a framework that is tweaked for LI. For other applications, the framework can be extended, for example, by specifying more categories of identifiers or definitions of other operations.

An earlier version of the proposed identity graphs was implemented as a part of the Sec6Net project [152, 159] based on my idea of identity linking in graphs. Formal description of identity graphs (as implemented as by the Sec6Net project) was published in the paper *On Identities in Modern Networks* [157]. I am the author of the text describing identity graphs and I developed the formulae in the formal description of identity graphs with a minor help of my co-workers. The implementation of identity graphs, as described in this thesis, is available on GitHub[1]. Compared to the original model, identity graphs defined in this thesis (1) have a notion of time, (2) allow better handling of inaccurate partial identity detectors (such as clock-skew-based identification described in Chapter 7), (3) allow additional custom operations – constraint functions, and (4) support identifiers of resources such as chat rooms.

Identity graphs are generic and they support identifiers that appear in all layers of the TCP/IP model (see Chapter 3). As a part of SLIS developed by the Sec6Net project, identity graphs were validated for identity linking based on the following partial identity detectors: DHCP [49], DHCPv6 [19], ND [136] (see Chapter 6), RADIUS [165], PPPoE [120], XMPP [169], IRC [106, 107], OSCAR (proprietary protocol for instant messaging), YMSG (proprietary protocol for instant messaging), SMTP [111], clock skew (see Chapter 7), and Software Defined Networking controllers — OpenDaylight[2] and Pox[3].

8.1 Related work in identity linking

Chapter 4 describes current standards of LI. For the European Union, the most relevant documents are ETSI standards. As mentioned in Chapter 4, the linkage has to be unambiguous. This chapter defines identity graphs that support various identifiers including those listed by ETSI [58]. In addition to unambiguous partial identity detectors, identity graphs also support inaccurate partial identity detectors. However, inaccurate partial detectors are not mandatory and the operations allow penalisation of the linkage discovered by inaccurate partial identity detectors. Also, linkage using information based on inaccurate partial identity detectors can be completely disabled.

Digital forensic literature, for example, Casey [26, page 650–651], emphasise the need to link identities and to select identifiers of the traffic to be intercepted based on the specific network parameters. Identity graphs allow custom queries with arbitrary input identifier and constraints. Casey also mentions that for some intercepts, investigators are authorised to monitor only a specific traffic, for example, web. Identity graphs allow interception of both (1) a single session identified by an application layer identifier and (2) all traffic of the computer where a user authenticates using the same application layer identifier. The distinction between these two use cases can be applied on a per intercept basis. Identity graphs allow setting specific constraints for each query. Consequently, it is possible to link identifiers of different intercepts based on different rules.

Casey [26, Section 22.8] describes a final report of a digital investigation. Casey proposes to create diagrams of complex interactions between computers connected to a network.

[1]https://github.com/polcak/linking
[2]https://www.opendaylight.org/
[3]https://openflow.stanford.edu/display/ONL/POX+Wiki

Indeed, identity graphs are graph diagrams of the identifiers in the network. Consequently, identity graphs can be used by network forensic experts during the writing of final reports.

Casey [26] describes log files that are available on various devices in the network including DHCP servers, RADIUS servers, ARP tables of network devices. Identity graphs support multiple diverse partial identity detectors including log file analysers. The extensibility of the mechanisms that builds identity graphs allows development of additional partial identity detectors.

Casey [26, section 3.3] considers digital investigation with estimated levels of certainty. Hence, Casey considers the digital evidence with non-zero inaccuracy. Casey argues that the system should evaluate the data and the probability of linking the identities correctly. Identity graphs support partial identity detectors that estimate their inaccuracy. The identifiers are linked based on the inaccuracy.

Hoffman and Terplan [90, pages 27, 33, 45] describe that there are many technologies in the network environment which require a mechanism that links identities. Additionally, Hoffman and Terplan [90, page 176] note: "Unfortunately, there are very few known products on the market that support data mining, evidence finding, and correlation functions". Identity graphs aim to fill the gap in identity linking and correlation.

Carmagnola et al. [25] proposed a system that crawls social networking websites. For each detected partial identity on a website, they look for so-called *replicated public data* — common information available on different profiles of the same real person. Based on the replicated public data they link the profiles — partial identities of the subject. Replicated public data have the same role as identifiers detected by different partial identity detectors investigated during this PhD research. The difference lays in the different domains investigated by Carmagnola et al. and this PhD research. This PhD research targets data transferred via network whereas Carmagnola et al. studied data stored on a website.

In principle, the crawler of Carmagnola [25] or a related work [131, 143] can be deployed as a partial identity detector based on which it is possible to construct an identity graph. However, identity graphs and operations defined in Section 8.4 form a framework that includes identifiers transferred through the network. The operations are not tweaked for attributes gathered from social networks. Nevertheless, the definitions can be extended to support social networks.

Torres et al. [185] studied identity management systems. As one of the key aspects, they list law enforcement interoperability. A modern LI system needs to process many input partial identity detectors. Identity graphs are based on this observation.

8.2 Detection of partial identities

Partial identities can be detected from many sources distributed in the network. As Chapter 3 established, protocols on different levels of network model carry various identifiers. Nevertheless, these identifiers do not necessarily identify the same subject. This section focuses on subjects of the identifiers and the detection of identifiers.

Partial identity detectors as traffic parsers

The activity of the Sec6Net project focusing on identification in LI (under my supervision) developed SLIS IRI-IIF modules that detect linkage between identifiers. A Sec6Net IRI-IIF module is a partial identity detector that monitors and analyses network traffic [159]. The implemented IRI-IIF modules are listed below:

- The DHCP module links two identifiers that identify the same network interface — a MAC address and an assigned IPv4 address. Additionally, if a DHCP client presents DHCPClientID (identifying the computer) to the server, the DHCP module links the DHCPClientID to both the MAC address and the assigned IPv4 address.

- The DHCPv6 module links a DUID identifying a computer and the assigned IPv6 address (identity association for non-temporary addresses [19]) or an IPv6 prefix (prefix delegation [186]).

- The RADIUS module links a MAC address identifying a computer network interface to a RADIUS username identifying a user. In case that an IPv4 or IPv6 address is assigned in the Access-Accept message [165], the RADIUS module also links all assigned IP addresses to the MAC address and RADIUS username. Note that depending on the network, a RADIUS username identifies either a specific user (for example, in enterprise networks) or a household (for example, when used by DSL provider).

- The PPPoE module links a MAC address identifying a PPP client interface to a PPP username identifying a user. When an IPv4 or IPv6 address is assigned to the PPPoE client, the PPPoE module links the IP address to other detected identifiers.

- The XMPP module links an XMPP username (also known as Jabber ID) to a TCP bi-directional flow carrying a session. Additionally, the module detects the IP address of the XMPP client. In some cases, such as a conversation between two XMPP subscribers, the XMPP module also detects the Jabber ID of the other communicating party.

- The IRC module links an IRC username to the TCP bi-directional flow carrying a session. Additionally, the module detects the IP address of the IRC client. Moreover, the IRC module detects IRC channels (essentially chat rooms) that the users enter.

- The OSCAR module links an OSCAR ID to the bi-directional flow carrying a session. Additionally, the module detects the IP address of the OSCAR client. In some cases, such as a conversation between two subscribers, the OSCAR module detects the OSCAR ID of the other communicating party.

- The YMSG module links a YMSG username to the bi-directional flow carrying a session. Additionally, the module detects the IP address of the YMSG client. In some cases, such as a conversation between two subscribers, the YMSG module detects the YMSG username of the other communicating party.

- The SMTP module links a bi-directional flow to e-mail addresses listed in the e-mail headers carried in the flow.

Chapters 6 and 7 presented two traffic analysing methods in detail. The IPv6 address assignment tracking proposed in Chapter 6 provides a linkage between a MAC address and an IPv6 address. The clock-skew-based remote identification described in Chapter 7 aims at linking two or more IP addresses belonging to the same computer based on a similar clock skew estimation. However, as presented in Chapter 7, the clock-skew-based identification is not completely accurate as it can identify an anonymity set containing multiple subjects instead of a single subject.

Partial identity detectors as log file analysers

Another option for identity-related information discovery is the log file analysis [26, Chapter 25]. Many computer services create log files or Syslog messages [72] from which it is possible to reconstruct and link identities related to the service.

Figure 8.1 shows a 6-line excerpt of the Apache web server[4]. Each line represents an attempt to access a resource. Each line provides the IP address initiating the request and the resource URI. The IP address identifies the client and the URI identifies the resource that the client wanted to visit.

```
147.126.81.180 - - [01/Mar/2017:10:15:17 +0100] "GET /robots.txt HTTP/1.1" 200 26 "-" "CSS Certificate Spider (http://www.css-security.com/certificatespider/)"
149.172.126.143 - - [01/Mar/2017:10:39:06 +0100] "GET / HTTP/1.1" 200 103 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36"
72.220.86.39 - - [01/Mar/2017:15:25:15 +0100] "GET / HTTP/1.1" 403 209 "-" "-"
103.109.210.141 - - [01/Mar/2017:11:02:56 +0100] "GET /status HTTP/1.1" 403 215 "-" "Mozilla"
103.109.210.141 - - [01/Mar/2017:13:45:33 +0100] "GET /stat HTTP/1.1" 403 213 "-" "Mozilla"
45.195.61.166 - - [01/Mar/2017:15:42:05 +0100] "GET / HTTP/1.1" 403 209 "-" "Lynx/2.8.8dev.3 libwww-FM/2.14 SSL-MM/1.4.1"
```

Figure 8.1: An example of an Apache log file.

Figure 8.2 shows an example of two DHCP assignments. Firstly, a DHCP client with ID Calgary running on a computer having network interface with MAC address *fc:99:47:cb:d9:68* leases IP address *10.10.10.207*. Later, a DHCP client with ID Ohio leases IP address *10.10.10.218* for interface with MAC address *cc:ef:48:e5:d6:e0*.

```
Mar 10 09:34:21 isa dhcpd: DHCPDISCOVER from fc:99:47:cb:d9:68 via em0
Mar 10 09:34:22 isa dhcpd: DHCPOFFER on 10.10.10.207 to fc:99:47:cb:d9:68 (Calgary) via em0
Mar 10 09:34:22 isa dhcpd: DHCPREQUEST for 10.10.10.207 (10.10.10.1) from fc:99:47:cb:d9:68 (Calgary) via em0
Mar 10 09:34:22 isa dhcpd: DHCPACK on 10.10.10.207 to fc:99:47:cb:d9:68 (Calgary) via em0
Mar 10 10:48:45 isa dhcpd: DHCPDISCOVER from cc:ef:48:e5:d6:e0 via em0
Mar 10 10:48:46 isa dhcpd: DHCPOFFER on 10.10.10.218 to cc:ef:48:e5:d6:e0 (Ohio) via em0
Mar 10 10:48:46 isa dhcpd: DHCPREQUEST for 10.10.10.218 (10.10.10.1) from cc:ef:48:e5:d6:e0 (Ohio) via em0
Mar 10 10:48:46 isa dhcpd: DHCPACK on 10.10.10.218 to cc:ef:48:e5:d6:e0 (Ohio) via em0
```

Figure 8.2: An example of a DHCP log file.

[4]https://httpd.apache.org/

Partial identity detectors as program extensions

Extensions (or plug-ins) for network-related programs are another option to get identity-related information. For example, the Sec6Net project developed plug-ins for OpenDaylight and Pox controller under my supervision [159]. For both controllers, the plug-in extracts a MAC address, IP address, switch ID, and switch port. The switch ID and switch port together identify a specific network interface (access point) that can (depending on the network structure) identify a user, a set of users, or a household.

Closing remarks on identity information sources

Indisputably, there are many sources of identity-related information. This thesis does not focus on provisioning an exhaustive list of partial identity detectors. Instead, this chapter provides operations that link identifiers learnt from different partial identity detectors deployed in the network.

Obviously, traffic analysis methods are applicable only to unencrypted traffic, or when encryption keys are available to partial identity detectors. Both log file analysis and program extensions can be deployed even when the traffic is encrypted. Log files contain the identifiers unencrypted and the program extensions have access to unencrypted identifiers in the internals of the network-related programs (IMSes).

Information provided by partial identity detectors

An identity graph incorporates identity-related information revealed by multiple partial identity detectors. A goal of each partial identity detector is to provide a connection between two or more identifiers; each identifier represents a different partial identity. Note that all detected identifiers can represent partial identities of the same subject. Typically, a partial identity detector supports a single method to reveal the linkage between identifiers.

As the challenge in Section 5.2 shows and Figures 8.1 and 8.2 confirms, network-related events happen over time, for example, temporary IPv6 addresses and DHCP leases are valid only for a specific period, visitors download web pages at specific moments. Hence, identity graphs incorporate time $\mathcal{T}$. This thesis considers ordered time values. Additionally, we suppose that $\mathcal{T}$ includes its supremum that we denote as ∞. As noted during the description of identity graphs, ∞ represents an unknown time in the future. Hence, $\mathcal{R} \cup \{\infty\}$ or ISO POSIX time are valid time representations.

Partial identity detectors produce messages reflecting changes in observed partial identities; each message contains:

1. Timestamp $t \in \mathcal{T}$ of the message.
2. The identifiers that the partial identity detector revealed are linked.
3. The information about the events concerning the linkage; possible events are *begin*, *continue*, or *end*.
4. The inaccuracy of the linkage. A *begin* message contains the initial inaccuracy, a *continue* message can update the inaccuracy. In this thesis, positive real numbers form the domain of the inaccuracy ($\mathit{Inaccuracy} \equiv \mathcal{R}_0^+$).

Note that the messages allow transformation into IRI records (*IRI begin*, *IRI continue*, *IRI end*) that are passed to LEA, see Chapters 4 and 9.

Accurate partial identity detectors, such as log parsers, advertise inaccuracy of 0. Other methods advertise the inaccuracy depending on the actual network state and configuration. In this thesis, we do not assume any particular meaning to specific values of inaccuracy except that higher values represent higher inaccuracy.

8.3 Categories of detected identifiers

Section 8.2 exposes that identifiers have different durability, and identify different types of subjects. This section investigates the differences. The goal of this section is to identify categories that are later applied in the operations for identity graphs.

Table 8.1 provides a list of identifiers mentioned in Section 8.2, their typical durability, and the subject which the identifier identifies. Note, that the table lists typical values. In some use cases, the behaviour differs, for example, recent Apple and Android mobile devices randomise MAC addresses. Final LI system deployment has to consider such nuances.

Table 8.1: Network identifiers discussed in Section 8.2.

Indentifier	**Durability**	**Identified subject**
MAC address	Typically long-term	Computer network interface
IPv6 address	Typically short-term	Computer network interface
IPv4 address	Dynamic	Computer network interface
DHCP client ID	Typically long-term	Computer
DHCPv6 DUID	Typically long-term	Computer (a single operating system)
RADIUS username	Long-term	User or household
PPP username	Long-term	User or household
TCP or UDP flow ID	Short-term	Session
XMPP username	Long-term	User
IRC username	Typically long-term	User
IRC channel name	Typically long-term	Chat room
OSCAR username	Long-term	User
YMSG username	Long-term	User
E-mail address	Long-term	User
URI	Dynamic	Web resource
Switch ID and port ID	Typically long-term	Computer, household

Some identifiers in Table 8.1 identify specific users, either on the application layer (for example, XMPP username, IRC username, OSCAR username, YMSG username, e-mail address) or for authentication purposes (for example, RADIUS, PPP). This PhD research treats these two groups differently:

- The application layer identifiers typically occur within a TCP or UDP session. Then, a partial identity detector identifies the linkage between an application username and a 5-tuple of transport protocol, client IP and port, and server IP and port.

  In some cases, such as SIP calls or FTP, a single session can be split into several TCP or UDP flows. The content of SIP calls is typically encapsulated in RTP. An FTP session consists of (1) a control channel, in which commands and replies are exchanged, and (2) multiple data channels, some commands open a data channel for the content of the reply, such as file transfer or directory listing. Identity graphs allow linkage between a single application layer identifier and multiple TCP and UDP 5-tuples.

- An authentication protocol typically authenticates a linkage between an authentication identifier and a network or a link layer identifier. Both network layer identifiers (typically IP addresses) and link layer identifiers (for example, MAC addresses) can be linked to other identifiers by the partial identity detectors.

Table 8.1 differentiate between identifiers of a computer and a network interface. However, in the context of LI, there is no difference between identifying an interface and a computer. Hence identity graphs do not treat identifiers of a network interface and identifiers of a computer differently.

Based on these observations, this thesis defines the following categories of identifiers:

L4Flow — Flows defined by a 5-tuple consisting of:

- transport layer protocol identifier, for example, TCP or UDP,
- client IP address and transport layer port number,
- server IP address and transport layer port number.

An *L4Flow* identifier identifies a transport layer flow. Every packet of the flow carries the identifier in header fields. LI probes developed as a part of the Sec6Net project can intercept traffic based on *L4Flow* identifiers.

IPAddr — IP addresses (typically short-term duration, assigned dynamically) identify an interface of a network node. Every IP packet carries source and destination IP addresses. LI probes developed as a part of the Sec6Net project can intercept traffic based on *IPAddr* identifiers.

IfcOrComp — Long-term identifiers of computers or network interfaces, such as MAC addresses, DUIDs, clock skew values. For identification methods that identify a specific resource on a computer, such as the browser fingerprinting [24, 51, 117, 128], the computer identifier reported by the identification method is an *IfcOrComp* identifier.

AAAUser — Authentication usernames of protocols such as RADIUS, PPP identify a set of network devices controlled by a unique user or a household depending on the network. Category *AAAUser* identifiers are transferred over the network only during an authentication phase of an authentication protocol.

L7User — Application layer usernames (for example, login names, account identifiers, e-mail addresses) identify a partial identity of a unique user. Usually, application layer usernames appear at least once in each session of an application layer protocol, however, such session may be composed of several transport layer flows (for example, SIP, FTP).

L7Resource — An application layer resource such as a chat room or a web page. Category *L7Resource* identifiers identify all application layer IOI except users. An *L7Resource* identifier is typically linked by an identification method to one or more *L7User* identifiers or *L4Flow* identifiers.

Let us define a set of categories as $Categories \equiv \{L4Flow, IPAddr, IfcOrComp, AAAUser, L7User, L7Resource\}$.

Note that there are two groups of identifiers of computers or network interfaces: *IPAddr* and *IfcOrComp*. For some readers, the division between *IPAddr* and *IfcOrComp* identifiers might seem to be redundant. We decided to make the division for the following reasons:

- It is common that a partial identity detector reveals a linkage between an *IPAddr* and an *IfcOrComp* identifier.
- It is not common that a partial identity detector reveals a linkage between two *IPAddr* identifiers or two *IfcOrComp* identifiers.
- The expected validity of an *IfcOrComp* identifier is longer compared to an *IPAddr* identifier.
- Category *IPAddr* identifiers appear in each packet.
- The distinction simplifies definitions of operations for identity graphs.

For the implemented partial identity detectors and during the exploration of other possible partial identity detectors, this PhD research empirically established the following rules for linked identities announced by partial identity detectors:

- An announced linkage between an *AAAUser* identifier and one or several *IPAddr* or *IfcOrComp* identifiers means that the *AAAUser* identifier authenticates the announced *IPAddr* and *IfcOrComp* identifiers.
- A single partial identity detector can link a single computer network interface identified by an *IfcOrComp* identifier to multiple *IPAddr* identifiers (for example, an interface identified by a MAC address configures multiple IPv6 addresses, see Chapter 6).
- A single partial identity detector can link a single computer identifier of category *IfcOrComp*, such as DUID or a fingerprint (for example, an estimated clock skew value or a browser fingerprint), to multiple *IPAddr* identifiers (for example, Chapter 7 describes linkage of IPv6 addresses of the same computer because of the similar clock skew detection).
- A single category *IPAddr* IP address can open multiple category *L4Flow* flows. This is announced as the linkage between the *IPAddr* and the *L4Flow* identifier.
- A user identified by an *L7User* identifier can initiate multiple sessions, each can be distributed in multiple flows identified by *L4Flow* identifiers. This is announced as multiple linkages, each linkage between the *L7User* and a single detected *L4Flow* identifier.
- Often, it is possible to link a user identified by an *L7User* identifier with the IP address of the computer (or network address translator) of the user. This is announced as the linkage of the *L7User* and the *IPAddr* identifier.
- A resource identified by an *L7Resource* identifier can be accessed in multiple flows, each identified by an *L4Flow* identifier. Additionally, the computer (identified by an *IPAddr* identifier) initiating the request for the resource can be linked to the *L7Resource* identifier. For some protocols, the resource can be linked to a user identified by an *L7User* identifier (for example, an IRC user enters a specific chatroom).

- NAT mapping between a flow identified by an *L4Flow* identifier (5-tuple) to another *L4Flow* identifier is announced as a linkage between the two *L4Flow* identifiers.

- For tunnelled traffic, it is necessary to detect the innermost identifiers in each packet as they identify the communicating parties and not tunnel endpoints.

8.4 Identity graph definition

This section specifies a model that validates Hypothesis 3 of this PhD research. The model employs the set *Categories*, the *Inaccuracy*, time $\mathcal{T}$, and the set *Detectors* of names of partial identity detectors. The set *Detectors* is finite in a particular deployment (as the number of partial identity detectors is finite in each deployment). Each detector advertises a linkage as specified in Section 8.2.

Definition 8.1. An **identity graph** is an extended multigraph $G \equiv (V, E, \mathit{endpts}, \mathit{category}, \mathit{attributes})$ where:

- (V, E, endpts) is multigraph, where:
  - V is a set of vertices. Every vertex represents a network identifier.
  - E is a set of edges.
  - endpts: $E \rightarrow \{\{x, y\} : x, y \in V \wedge x \neq y\}$ assigns each edge a pair of different vertices (endpoints).

- $\mathit{category}$: $V \rightarrow \mathit{Categories}$ is a total function that maps each vertex to its category from the set *Categories*.

- $\mathit{attributes}$: $E \rightarrow \mathit{Detectors} \times \mathcal{T} \times \mathcal{T} \times \mathit{Inaccuracy}$ is a total function that defines attributes of the edges. For each edge, function *attributes* stores its partial identity detector, the starting time and end time of the linkage, and the estimated inaccuracy of the identification method with respect to this linkage.

Function *attributes* maps an edge to a 4-tuple of attributes. However, for some operations, it is beneficial to access one of the attributes by projection Π — $\Pi_n(\mathit{attributes}(e))$ for $n \in \{1, 2, 3, 4\}$ and $e \in E$ refers to the n-th element of $\mathit{attributes}(e)$. For example, if $\mathit{attributes}(e) = \{d, t_0, t_e, i\}$, $\Pi_4(\mathit{attributes}(e)) = i$.

An identity graph is constructed from messages received from partial identity detectors based on algorithms 8.1, 8.2, and 8.3. As established in Section 8.2, three types of messages can be sent from a partial identity detector $d \in \mathit{Detectors}$ (each message carries a set of identifiers I and a timestamp $t \in \mathcal{T}$. Additionally, *begin* and *continue* carry inaccuracy $i \in \mathit{Inaccuracy}$). Appendix B provides examples of the application of Algorithms 8.1, 8.2, and 8.3.

Algorithm 8.1. Handling of *begin* messages: For each *begin* received from any detector d:

1. Insert previously unseen identifiers to V ($V := V \cup I$).

2. Insert a new edge e to the set E for each pair of identifiers $u, v \in I$. Redefine *endpts* so that for each new $e \in E$ added in this step, $\mathit{endpts}(e) \equiv \{u, v\}$ ($u \neq v$; u, v represents the vertices of the identifiers linked by the edge).

3. Redefine *category* so that for each $i \in I$, $\mathit{category}(i)$ returns the category of i.

4. Redefine *attributes* so that for each new e, $\mathit{attributes}(e) := (d, t, \infty, i)$.

5. Additionally, remember the set of edges added by the *begin* message as they can be updated by subsequent *continue* and *end* messages.

Algorithm 8.2. Handling of *continue* messages: For each *continue* message update the inaccuracy. Find edges inserted by the corresponding *begin* message. For each edge e, if $i \neq \Pi_4(\mathit{attributes}((e))$:

1. Redefine *attributes* so that $\Pi_3(\mathit{attributes}(e)) \equiv t$.

2. Insert a new edge e_c to E. Redefine *endpts* so that $\mathit{endpts}(e_c) \equiv \mathit{endpts}(e)$.

3. Redefine *attributes* so that $\mathit{attributes}(e_c) := (d, t, \infty, i)$.

4. Remember that the new edge e_c should be updated by subsequent *continue* and *end* messages instead of the original edge e.

Algorithm 8.3. Handling of *end* messages: For each *end* message, find edges inserted by the corresponding *begin* and *continue* messages. For each edge e that was not updated by any *continue* message, that is $\Pi_3(\mathit{attributes}(e)) = \infty$, redefine *attributes* so that $\Pi_3(\mathit{attributes}(e)) \equiv t$.

Queries in identity graphs are based on paths in the graph, a path is a specific type of walk. Definition 8.2 defines a walk [79] and Definition 8.3 defines paths [79].

Definition 8.2. A **walk** [79] from vertex v_0 to vertex v_n is an alternating sequence $\langle v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n \rangle$ of vertices and edges, such that $\mathit{endpts}(e_i) = \{v_{i-1}, v_i\}$ and $e_i \in E$ for $i = 1 \ldots n$.

Definition 8.3. A **path** [79] is a walk with neither repeated vertices, nor repeated edges.

In formulae where vertices are not important, vertices are omitted from the representation of the path sequence, for example, $\langle e_1, e_2, \ldots, e_{n-1}, e_n \rangle$ [79].

Let us denote the set of all paths in an identity graph G as $\mathcal{P}(G)$.

The linking operations in the identity graph defined below employ constraint functions (predicates) as defined in Definition 8.4.

Definition 8.4. A **constraint function** $L\colon \mathcal{P}(G) \to \{\text{true}, \text{false}\}$ evaluates a path and yields *true* if the path fulfils the requirements given by the constraint and *false* otherwise.

Formula 8.1 defines the generic form of the function *linked* that yields a set of linked identifiers for an input identifier represented by vertex $v_i \in V$ based on a set of constraint functions L_f. Examples of constraint functions are provided later in this chapter.

$$\begin{gathered}\mathit{linked}(v_i, L_f, G) \equiv \Bigg\{ o \in V : \\ (\exists \langle v_0, e_1, \ldots, e_n, v_n \rangle \in \mathcal{P}(G)) : v_0 = v_i \wedge v_n = o \wedge \left(\bigwedge_{f \in L_f} (f(\langle v_0, e_1, \ldots, e_n, v_n \rangle)) \right) \Bigg\}\end{gathered} \tag{8.1}$$

Examples of constraint functions follow. Note that Appendix B provides examples of applications of operations in identity graphs based on the following constraint functions.

8.4.1 Constraint functions restricting relations between identifiers

As mentioned in Section 5.5, the wording of a warrant for LI can influence the scope of traffic to be intercepted and the identifiers for which to generate IRI.

To address the challenge of allowed identifier linking, this thesis defines the following constraint functions $l_{8.3}$, $l_{8.5}$, $l_{8.7}$, $l_{8.8}$, $l_{8.10}$, and $l_{8.12}$ that restrict identifiers allowed on accepted paths based on the categories of identifiers on paths. Consequently, these constraint functions reveal the identifiers linked to the input identifier according to the wording of the warrant. Each constraint function is described in a subsubsection below.

Constraints revealing components of partial identity

Sometimes an LI warrant lists an identifier A that should be monitored. However, the partial identity represented by the identifier A consists of several partial subidentities with their specific identifiers; each partial subidentity is a subset of the original partial identity.

For example, consider a warrant aiming at the CC interception of data linkable to a DHCPv6 DUID. The intercept should cover all IP addresses assigned for the DHCPv6 DUID, but it must not cover other IP addresses even if they are assigned to the same interface.

Suppose that vertex v_0 represents the identifier A (listed on the warrant). The goal is to find all related identifiers that are represented by vertices on paths starting in v_0 have the following constraints (notice that the edges can be on the path only in specified directions) empirically specified during this PhD research:

- An edge from an *AAAUser* identifier to an *IfcOrComp* or *IPAddr* identifier is allowed because the *IfcOrComp* and *IPAddr* identifiers are authenticated by the *AAAUser* identifier.
- An edge from an *IfcOrComp* identifier to an *IPAddr* identifier is allowed because the IP address (*IPAddr* identifier) belongs to the interface or a computer (*IfcOrComp* identifier).
- An edge from an *IPAddr* identifier to an *L4Flow* identifier is allowed because the IP address (*IPAddr* identifier) communicates in the flow (*L4Flow* identifier).
- Note that an edge from an *IPAddr* identifier to an *L7User* identifier is not allowed as the *L7User* identifier can also be used by partial identities identified by different *IPAddr* identifiers. For example, the same user can log in at different computers.
- An edge from an *L7User* identifier to an *L4Flow* identifier is allowed because the user (*L7User* identifier) communicates in the flow (*L4Flow* identifier).
- An edge from an *L4Flow* identifier to another *L4Flow* identifier reflects a NAT mapping. Hence, both identifiers identify the same flow.

Relation $r_{8.2} \subseteq \mathit{Categories} \times \mathit{Categories}$ represents the constraints specified above as

$$\begin{aligned} r_{8.2} \equiv \{&(\mathit{L4Flow}, \mathit{L4Flow}), (\mathit{IPAddr}, \mathit{L4Flow}), (\mathit{IfcOrComp}, \mathit{IPAddr}), \\ &(\mathit{AAAUser}, \mathit{IPAddr}), (\mathit{AAAUser}, \mathit{IfcOrComp}), (\mathit{L7User}, \mathit{L4Flow})\}. \end{aligned} \tag{8.2}$$

Formula 8.3 defines constraint function $l_{8.3}$ applied for a path $p = \langle v_0, e_1, v_1, \ldots, e_n, v_n \rangle$. Constraint function $l_{8.3}$ allows only paths that represent components of a partial identity represented by v_0.

$$l_{8.3}(\langle v_0, e_1, v_1, \ldots, e_n, v_n \rangle) \equiv \begin{cases} \text{true} & \text{if } ((\forall i \in [1, n]) : r_{8.2}(\mathit{category}(v_{i-1}), \\ & \quad \mathit{category}(v_i))), \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.3}$$

Figure 8.3 displays a diagram that visualises categories of identifiers on paths allowed by Formula 8.3. Only paths of two or more nodes are accepted.

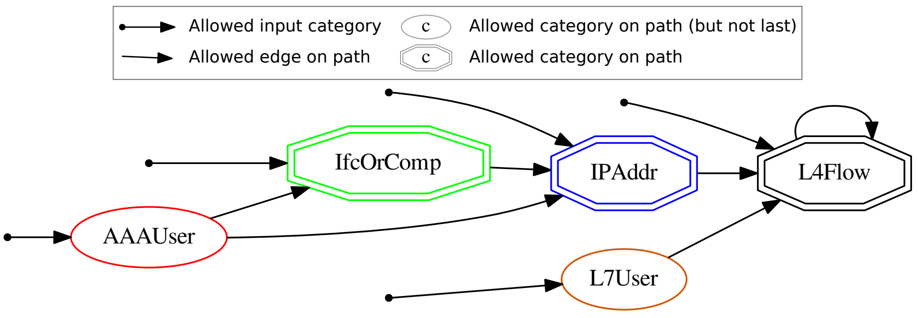


Figure 8.3: Categories of identifiers on paths accepted by Formula 8.3.

Constraints revealing partial identities of specific computer

Sometimes an LI warrant aims at intercepting all traffic of the computer identified by the identifier A. Hence, all partial identities that are linkable to be used on the same computer are covered by the intercept. Such LI warrant is only defined if the identifier A is of the category *IPAddr* or *IfcOrComp* as identifiers of these categories represent a computer or its network interface.

For example, consider a warrant for interception of the traffic of a computer identified by a DHCPv6 DUID. In this case, the intercept covers all IP addresses assigned for the DHCPv6 DUID as well as all other addresses configured on the computer (on the same interface and all other interfaces).

Suppose that vertex v_0 represents the identifier A listed on the warrant. The goal is to find all linkable identifiers that are present at the same computer and cannot be present on other computers. Hence, all identifiers on the path from v_0 to a linked identifier have to belong to the same computer. The path has the following constraints (notice that the edges can be on the path only in the specified directions) empirically specified during this PhD research:

- An edge from an *IfcOrComp* identifier to an *IPAddr* identifier is allowed in both directions because the IP address (*IPAddr* identifier) belongs to the interface or a computer (*IfcOrComp* identifier).
- An edge from an *IPAddr* identifier to an *L4Flow* identifier is allowed because the IP address (*IPAddr* identifier) communicates in the flow (*L4Flow* identifier).

- An edge from an *L4Flow* identifier to another *L4Flow* identifier reflects a NAT mapping. Hence, both identifiers identify the same flow and the edge connect two vertices representing identifiers of traffic of the same computer.

Relation $r_{8.4} \subseteq \mathit{Categories} \times \mathit{Categories}$ represents the constraints specified above as

$$\begin{aligned} r_{8.4} \equiv \{&(\mathit{L4Flow}, \mathit{L4Flow}), (\mathit{IPAddr}, \mathit{L4Flow}),\\ &(\mathit{IPAddr}, \mathit{IfcOrComp}), (\mathit{IfcOrComp}, \mathit{IPAddr})\} \end{aligned} \tag{8.4}$$

Formula 8.5 defines constraint function $l_{8.5}$ applied for a path $p = \langle v_0, e_1, v_1, \ldots, e_n, v_n \rangle$. Constraint function $l_{8.5}$ allows only paths that represent a specific computer.

$$l_{8.5}(\langle v_0, e_1, v_1, \ldots, e_n, v_n \rangle) \equiv \begin{cases} \text{true} & \text{if } \mathit{category}(v_0) \in \{\mathit{IPAddr}, \mathit{IfcOrComp}\} \wedge \\ & \quad \wedge ((\forall i \in [1, n]) : r_{8.4}(\mathit{category}(v_{i-1}), \\ & \quad \mathit{category}(v_i))), \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.5}$$

Figure 8.4 displays a diagram that visualises categories of identifiers on paths allowed by Formula 8.5. Only paths of two or more nodes are accepted.

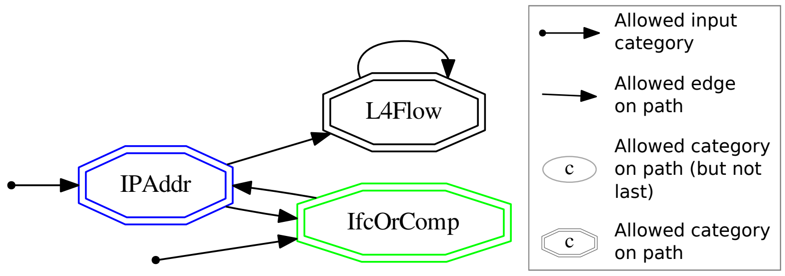


Figure 8.4: Categories of identifiers on paths accepted by Formula 8.5.

Constraints revealing partial identities of computers where specific user authenticated or logged in

An LI warrant that orders interception of all traffic of all computers authenticated by a specific user is applicable only if the input identifier is of the *AAAUser* or *L7User*.

An *AAAUser* identifier can be used for authentication of several computers. For example, a RADIUS username for *eduroam*[5] can authenticate a laptop, a phone, and a tablet of the same user simultaneously or over time. If A is an *AAAUser* identifier, then all identifiers linkable to all computers authenticated by A are covered by the warrant.

Similarly, a user can log in using an *L7User* identifier on several computers. Hence, the LI warrant for all computers traffic where a user has logged in covers all intercepts of all computers identified by IP addresses represented by vertices with edges to the vertex of the identifier A.

[5]https://www.eduroam.org/

Equivalence 8.6 defines the relation $r_{8.6} \subseteq \mathit{Categories} \times \mathit{Categories}$ that allows linkage (1) from an *AAAUser* identifier to *IfcOrComp* and *IPAddr* identifiers, (2) from an *L7User* identifier to *IPAddr* identifiers.

$$r_{8.6} \equiv \{(\mathit{AAAUser}, \mathit{IPAddr}), (\mathit{AAAUser}, \mathit{IfcOrComp}), (\mathit{L7User}, \mathit{IPAddr})\} \tag{8.6}$$

Formula 8.7 defines constraint function $l_{8.7}$ applicable for a path $p = \langle v_0, e_1, v_1, \ldots, e_n, v_n\rangle$. Constraint function $l_{8.7}$ allows only paths from a *AAAUser* or *L7User* identifiers traversing identifiers belonging to (1) computers authenticated by the identifier represented by v_0 of category *AAAUser* and (2) computers used by a user that accessed the account represented by the identifier of v_0 of category *L7User*.

$$l_{8.7}(\langle v_0, e_1, v_1, \ldots, e_n, v_n\rangle) \equiv \begin{cases} \text{true} & \text{if } r_{8.6}(\mathit{category}(v_0), \mathit{category}(v_1)) \wedge \\ & \quad \wedge((\forall i \in [2, n]) : r_{8.4}(\mathit{category}(v_{i-1}), \\ & \quad \mathit{category}(v_i))), \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.7}$$

Figure 8.5 displays a diagram that visualises categories of identifiers on paths allowed by Formula 8.7. All paths start either in an *AAAUser* or *L7User* identifier. The path can end in any identifier of the category *L4Flow*, *IPAddr*, or *IfcOrComp*.

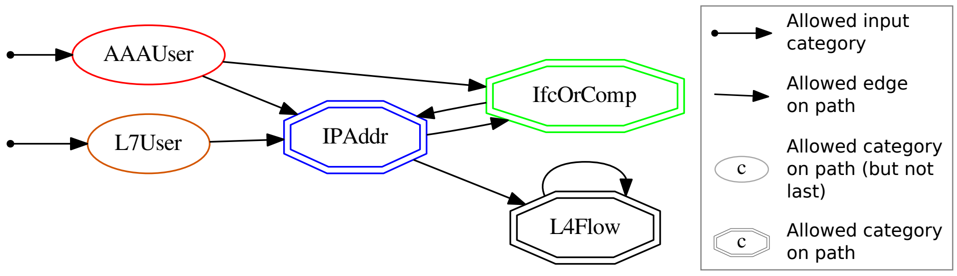


Figure 8.5: Categories of identifiers on paths accepted by Formula 8.7.

Constraints revealing identifiers of all users accessing specific resource

An LI warrant aiming at monitoring of a specific resource provides an *L7Resource* input identifier. The result is a set of the directly connected *L7User* identifiers.

Formula 8.8 defines the constraint function $l_{8.8}$ applicable for a path $p = \langle v_0, e_1, v_1, \ldots, e_n, v_n\rangle$. Constraint function $l_{8.8}$ allows only paths of two vertices from an *L7Resource* identifier to *L7User* identifiers.

$$l_{8.8}(\langle v_0, e_1, v_1, \ldots, e_n, v_n\rangle) \equiv \begin{cases} \text{true} & \text{if } (n = 1) \wedge \mathit{category}(v_0) = \mathit{L7Resource} \wedge, \\ & \wedge\, \mathit{category}(v_1) = \mathit{L7User} \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.8}$$

Figure 8.6 displays a diagram that visualises categories of identifiers on paths allowed by Formula 8.8. All paths start in an *L7Resource* identifier and finish in a directly connected *L7User* identifier.

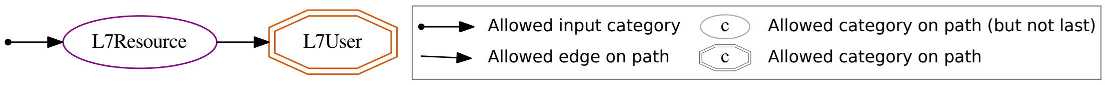


Figure 8.6: Categories of identifiers on paths accepted by Formula 8.8.

Constraints revealing all user accounts logged in or authenticated from computer or set of computers

An LI warrant for all user accounts accessed from a computer or a set of computers targets an *IPAddr*, *IfcOrComp*, *AAAUser* or *L7User* identifier A. The expected result is a set of identifiers of category *AAAUser* or *L7User*. Depending on the category of the input identifier the meaning of the intercept is slightly different:

- An *IPAddr* or an *IfcOrComp* identifier means that all user accounts logged in or authenticated from a computer identified by A should be linked.
- An *AAAUser* identifier means that all user accounts logged in or authenticated from all computers authenticated by A should be linked.
- An *L7User* identifier means that all user accounts logged in or authenticated from all computers from which the account A was accessed should be linked.

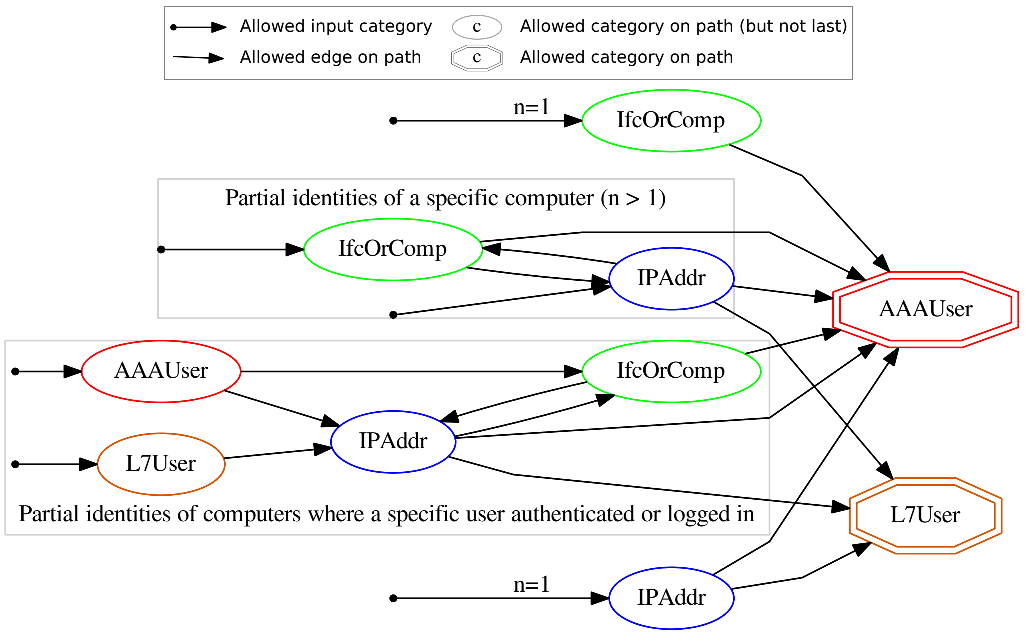
Equivalence 8.9 defines the relation $r_{8.9} \subseteq \mathit{Categories} \times \mathit{Categories}$ that allows linkage between (1) an IP address and the username accessed from that address or (2) an *IPAddr* or *IfcOrComp* identifier authenticated by an *AAAUser* identifier.

$$r_{8.9} \equiv \{(\mathit{IPAddr}, \mathit{L7User}), (\mathit{IPAddr}, \mathit{AAAUser}), (\mathit{IfcOrComp}, \mathit{AAAUser})\} \tag{8.9}$$

Formula 8.10 defines constraint function $l_{8.10}$ applicable for a path $p = \langle v_0, e_1, v_1, \ldots, e_n, v_n\rangle$. Constraint function $l_{8.10}$ allows linking identifiers of the partial identity of the devices (1) selected by the input identifier ($l_{8.5}$) or (2) where the user authenticated or logged in ($l_{8.7}$). Only paths ending by two identifiers of the categories accepted by $r_{8.9}$ are allowed by Formula 8.10.

$$l_{8.10}(\langle v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n\rangle) \equiv \begin{cases} \text{true} & \text{if } (l_{8.5}(\langle v_0, e_1, v_1, \ldots, e_{n-1}, v_{n-1}\rangle)\vee \\ & \quad l_{8.7}(\langle v_0, e_1, v_1, \ldots, e_{n-1}, v_{n-1}\rangle)\vee \\ & \quad n = 1) \wedge r_{8.9}(\mathit{category}(v_{n-1}), \\ & \quad \mathit{category}(v_n)) \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.10}$$

Figure 8.7 displays a diagram that visualises categories of identifiers on paths allowed by Formula 8.10. Each path starts in a vertex of a category *IPAddr*, *IfcOrComp*, *AAAUser*, or *L7User*. Accepted paths are at least of the length of two nodes and finish in one of the vertices representing identifiers of category *AAAUser* or *L7User*.



Allowed input category
Allowed edge on path
c
Allowed category on path (but not last)
c
Allowed category on path
n=1
IfcOrComp
Partial identities of a specific computer (n > 1)
IfcOrComp
IPAddr
AAAUser
AAAUser
IfcOrComp
IPAddr
L7User
Partial identities of computers where a specific user authenticated or logged in
L7User
n=1
IPAddr

Figure 8.7: Categories of identifiers on paths accepted by Formula 8.10.

Constraints revealing all accessed resources

An LI warrant for the accessed resources targets an *IPAddr*, *IfcOrComp*, *AAAUser* or *L7User* identifier A. Depending on the category of the input identifier the meaning of the intercept is slightly different:

- An *IPAddr* or an *IfcOrComp* identifier means that all resources accessed from a computer identified by A should be linked.
- An *AAAUser* identifier means that all resources accessed from all computers authenticated by A should be linked.
- An *L7User* identifier means that all resources accessed from all computers from which the account A was accessed should be linked.

Equivalence 8.11 defines the relation $r_{8.11} \subseteq \mathit{Categories} \times \mathit{Categories}$ that allows linkage between (1) an IP address and the resource accessed from that IP address, and (2) an application username and the resource accessed by the user.

$$r_{8.11} \equiv \{(\mathit{IPAddr}, \mathit{L7Resource}), (\mathit{L7User}, \mathit{L7Resource})\} \tag{8.11}$$

Formula 8.12 defines constraint function $l_{8.12}$ applicable for a path $p = \langle v_0, e_1, v_1, \ldots, e_{n-1}, v_{n-1}, e_n, v_n \rangle$. Constraint function $l_{8.12}$ allows only paths that end in a vertex representing an *L7Resource* identifier.

$$l_{8.12}(\langle v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n\rangle) \equiv \begin{cases} \text{true} & \text{if } (l_{8.5}(\langle v_0, e_1, v_1, \ldots, e_{n-1}, v_{n-1}\rangle) \vee \\ & \quad l_{8.7}(\langle v_0, e_1, v_1, \ldots, e_{n-1}, v_{n-1}\rangle) \vee \\ & \quad n = 1) \wedge r_{8.11}(\mathit{category}(v_{n-1}), \\ & \quad \mathit{category}(v_n)) \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.12}$$

Figure 8.8 displays a diagram that visualises categories of identifiers on paths allowed by Formula 8.12. Each path starts in a vertex representing an identifier of category *IPAddr*, *IfcOrComp*, *AAAUser*, and *L7User*. Accepted paths are at least of the length of two nodes and finish in one of the vertices representing category *L7Resource* identifiers.

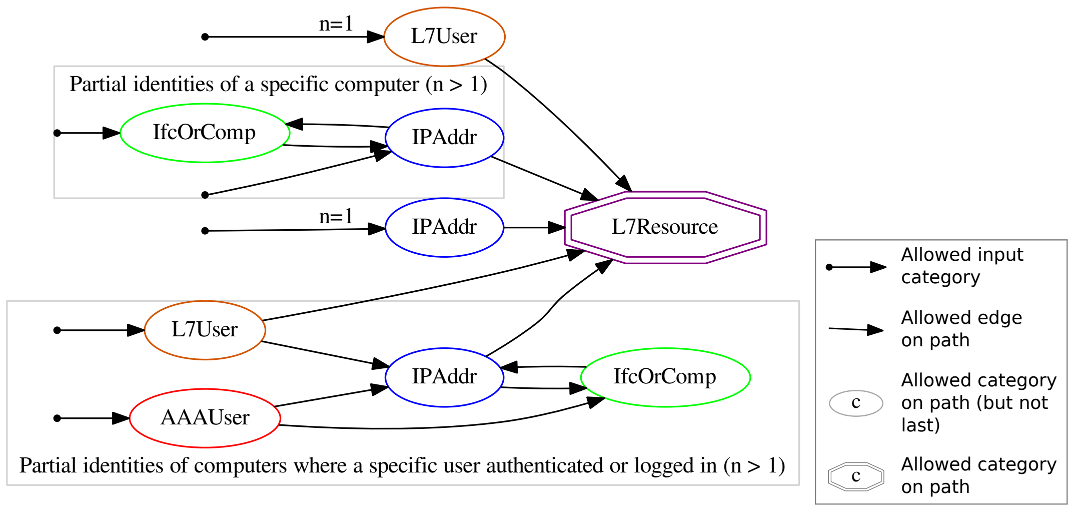

n=1
L7User
Partial identities of a specific computer (n > 1)
IfcOrComp
IPAddr
n=1
IPAddr
L7Resource
L7User
IPAddr
IfcOrComp
AAAUser
Partial identities of computers where a specific user authenticated or logged in (n > 1)
Allowed input category
Allowed edge on path
c
Allowed category on path (but not last)
c
Allowed category on path

Figure 8.8: Categories of identifiers on paths accepted by Formula 8.12.

8.4.2 Time constraints

In some cases, it is necessary to consider only connections in the graph that are valid at a certain time or in a time range, for example, last month, during a specific day, after a specific time point. Let $t_0, t_1 \in \mathcal{T}$ define a period ($t_1 > t_0$) or a specific moment ($t_0 = t_1$).

Formula 8.13 provides one example of a time-based constraint function template $l_{8.13}$: $\mathcal{T} \times \mathcal{T} \times \mathcal{P} \to \{\text{true}, \text{false}\}$. All edges on the path have to be valid during the whole period $[t_0, t_1]$.

$$l_{8.13}(t_0, t_1, \langle e_1, e_2, \ldots, e_n\rangle) \equiv \begin{cases} \text{true} & \text{if } ((\forall i \in [1, n]) : \Pi_2(\mathit{attributes}(e_i)) \leq t_0 \wedge \\ & \qquad \wedge \Pi_3(\mathit{attributes}(e_i)) \geq t_1), \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.13}$$

Formula 8.14 provides another example of a time-based constraint function template $l_{8.14}$: $\mathcal{T} \times \mathcal{T} \times \mathcal{P} \to \{\text{true}, \text{false}\}$. All edges on the path have to be valid at least once during the period $[t_0, t_1]$ and the period during the previous identifier is valid on the path.

$$l_{8.14}(t_0, t_1, \langle e_1, e_2, \ldots, e_n \rangle) \equiv \begin{cases} \text{true} & \text{if } [t_0, t_1] \cap [\Pi_2(\mathit{attributes}(e_1)), \\ & \quad \Pi_3(\mathit{attributes}(e_1))] \neq \emptyset \wedge ((\forall i \in [2, n])\colon \\ & \quad [t_0, t_1] \cap [\Pi_2(\mathit{attributes}(e_i)), \\ & \quad \Pi_3(\mathit{attributes}(e_i))] \cap [\Pi_2(\mathit{attributes}(e_{i-1})), \\ & \quad \Pi_3(\mathit{attributes}(e_{i-1}))] \neq \emptyset) \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.14}$$

Let $l_{8.13}(a, b)$ denote a partially bounded function $l_{8.13}(t_0, t_1, \langle e_1, e_2, \ldots, e_n \rangle)$ where t_0 is bounded to a and t_1 is bounded to b. Similarly, let $l_{8.14}(a, b)$ denote a partially bounded function $l_{8.14}(t_0, t_1, \langle e_1, e_2, \ldots, e_n \rangle)$ where t_0 is bounded to a and t_1 is bounded to b. Such partially bounded functions $l_{8.13}(a, b)\colon \mathcal{P}(G) \to \{\text{true}, \text{false}\}$ and $l_{8.14}(a, b)\colon \mathcal{P}(G) \to \{\text{true}, \text{false}\}$ can be used as constraint functions in Formula 8.1.

Time limitations are useful for both LI and network forensic. Hence, typically all linking through the function *linked* (see Formula 8.1) is performed with a time-related constraint function.

For complex queries such as the linkage of identifiers that were repeatedly linkable to an identifier represented by $v_0 \in V$ at several distinct time periods, it is possible to intersect the sets returned by *linked*. Let L_F be a set of constraint functions, and let T be a set of pairs (t_0, t_1), $t_0 < t_1$; t_0 is the starting time and t_1 is the end time of one of the queried period.

Formula 8.15 yields a set of vertices that are linkable to the input vertex v_0 with a set of constraint functions L_f at least at some moment during each provided period in T.

$$\mathit{linked}_{\text{repeatedly}}(v_0, T, L_f, G) \equiv \bigcap_{(t_0, t_1) \in T} \mathit{linked}(v_0, L_f \cup \{l_{8.14}(t_0, t_1)\}, G) \tag{8.15}$$

Similarly, Formula 8.16 merges sets of identifiers linkable to an identifier represented by $v_0 \in V$ at least at a single moment during at least one period in T.

$$\mathit{linked}_{\text{anytime}}(v_0, T, L_f, G) \equiv \bigcup_{(t_0, t_1) \in T} \mathit{linked}(v_0, L_f \cup \{l_{8.14}(t_0, t_1)\}, G) \tag{8.16}$$

8.4.3 Inaccuracy constraints

In case that some linking information is based on inaccurate partial identity detectors, for example, clock skew, browser fingerprinting, the linking is not transitive. Inaccuracy constraint functions limit the transitivity.

Function $l_{8.17} : \mathcal{R}_0^+ \times \mathcal{P}(G) \to \{\text{true}, \text{false}\}$ evaluates all edges for observed inaccuracy and allows paths $\langle e_1, e_2, \ldots e_n \rangle$ where each edge has inaccuracy lower than given threshold I_T, see Formula 8.17.

$$l_{8.17}(I_T, \langle e_1, e_2, \ldots, e_n \rangle) \equiv \begin{cases} \text{true} & \text{if } (\forall i \in [1, n] : \Pi_4(\mathit{attributes}(e_i)) \leq I_T), \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.17}$$

Formula 8.18 defines path inaccuracy function $\text{Inaccuracy}_\text{P} : \mathcal{P}(G) \to \mathcal{R}_0^+$ of a path as a sum of the inaccuracies detected on all edges along the path $\langle e_1, e_2, \ldots, e_n \rangle$.

$$\text{Inaccuracy}_{\text{P}}(\langle e_1, e_2, \ldots, e_n \rangle) \equiv \sum_{i=1}^{n} \Pi_4(\mathit{attributes}(e_i)) \tag{8.18}$$

Formula 8.19 defines constraint function template $l_{8.19} : \mathcal{R}_0^+ \times \mathcal{P}(G) \to \{\text{true}, \text{false}\}$ that compares the total inaccuracy along a path with a threshold I_T.

$$l_{8.19}(I_T, \langle e_1, e_2, \ldots, e_n \rangle) \equiv \begin{cases} \text{true} & \text{if } \text{Inaccuracy}_{\text{P}}(\langle e_1, e_2, \ldots, e_n \rangle) \leq I_T, \\ \text{false} & \text{otherwise.} \end{cases} \tag{8.19}$$

Let $L_{8.19}(n)$ denote a partially bounded function $l_{8.19}(I_T, \langle e_1, e_2, \ldots, e_n \rangle)$ where I_T is bounded to n. Similarly, let $L_{8.17}(m)$ denote a partially bounded function $l_{8.17}(I_T, \langle e_1, e_2, \ldots, e_n \rangle)$, where I_T is bounded to m. Such partially bounded functions $L_{8.19}(n) \colon \mathcal{P} \to \{\text{true}, \text{false}\}$ and $L_{8.17}(m) \colon \mathcal{P} \to \{\text{true}, \text{false}\}$ can be used as constraint functions in Formula 8.1.

8.4.4 Application of constraint functions

The list of the constraint functions presented above is not exclusive. It is possible to extend the framework and define additional constraint functions based on the application of identity graphs.

Typically, there is more than one constraint function employed at the same time. See Appendix B for examples of identity graphs and operations on identity graphs.

8.4.5 Identity linking in networks with address translation

For networks performing NAT, in case that CC interception is located after the translator, the translation mapping between the flows has to be included in the identity graph. Hence, a partial identity detector has to reveal the NAT mapping. The translation can be detected (1) by probes before and after the translator, for example, NAT mapping detector of Grégr [80, Chapter 4] or a log file analyser [173], (2) from logs generated by a network address translator or (3) by another mechanism, such as clock-skew-based linking [112].

For identity graphs constructed in networks with NAT, the following rules apply:

1. The graph cannot contain edges between the IP address of the translator p represented by node $p' \in V$ of category *IPAddr* and a node $a \in V$ of category *L7User* or *L7Resource*. For each removed edge between p' and a, an edge between a and the vertex representing the local identifier of category *IPAddr* that is translated to p has to be added.

2. The following edges have to be added when a local flow x (represented by vertex $x' \in V$) is translated to a public flow y (represented by vertex $y' \in V$):

   (a) an edge between vertices x' and y';

   (b) For each edge e between the vertex x' and another vertex $z \neq y'$ ($\mathit{endpts}(e) = \{x', z\}$) if $\mathit{category}(z) \in \{\mathit{L7User}, \mathit{L7Resource}\}$, an edge e' has to be added to E and *endpts* redefined so that $\mathit{endpts}(e') = \{y', z\}$ and *attributes* has to be redefined so that $\mathit{attributes}(e) = \mathit{attributes}(e')$.

(c) For each edge e between the vertex y' and another vertex $z \neq x'$ ($endpts(e) = \{y', z\}$) if $category(z) \in \{L7User, L7Resource\}$, an edge e' has to be added to E and *endpts* redefined so that $endpts(e') = \{x', z\}$ and *attributes* has to be redefined so that $attributes(e) = attributes(e')$.

With these rules, the previously defined operations can be performed in networks with NAT. In networks with CGN, it is necessary to deploy partial identity detectors that detect both translations. The same rules as in networks with a single translation apply.

8.5 Validation

Identity graphs described in this chapter were validated in several scenarios during the Sec6Net project and for this thesis.

Validation as a part of the Sec6Net project — Originally, identity graphs were developed for SLIS (the LI system developed as a part of the Sec6Net project) under my supervision. Although the Sec6Net identity graphs [159, Chapter 5] support only a part of the operations available in identity graphs as described by this thesis, the deployment in many networks in the Sec6Net project (see Appendix C) showed the usefulness of the idea behind identity linking in identity graphs. In each deployment, one of the important tasks was the validation of correct linkage. As a part of the Sec6Net project, identity graphs were presented to Czech LEA officers during several demo sessions.

The generality of the identity graphs was proven by many partial identity detectors. Under my supervision, the Sec6Net project developed tools that reveal identity-related information from ten network protocols, two SDN controllers, and from clock-skew-based identification.

Bachelor and diploma thesis employing identity graphs — I supervised a bachelor thesis [103] that developed an inaccurate partial identity detector that used HTTP headers to identify browser profiles and consequently the user. Additionally, I supervised a diploma thesis [173] that focused on tunnelling protocols and NAT. Finally, I supervised a diploma thesis [92] that developed an authentication web page that acts as a partial identity detector.

Validation of identity graphs defined by this thesis The application of the original identity graphs for inaccurate partial identity detectors (clock-skew-based identification and browser fingerprinting) revealed that some limits for transitivity of the linkability of identifiers are required. The constraint function defined by Formula 8.19 allows limiting linkability of identifiers learnt from inaccurate partial identity detectors. The following experiment evaluates the applicability of the inaccuracy in identity graphs.

The experiment simulates an IPv6-enabled network of a small internet service provider. The internet service provider has 20 customer that accesses the network with 112 devices during the simulated period. Each device authenticates its MAC address with a RADIUS username belonging to the customer, each device leases an IPv4 address and generates IPv6 address. Additionally, each device can generate privacy extension IPv6 addresses [135] at any moment. During the experiment, all address assignments were monitored and an identity graphs containing the assignments was created. Figure 8.9 shows the number of IPv4 and IPv6 addresses active in the network through the simulation. IPv4 and IPv6 addresses can be queried by evaluating $\mathit{linked}(R, \{l_{8.3}, l_{8.13}(t_e, t_e)\}, G)$ for each RADIUS username R at time t_e in the identity graph G.

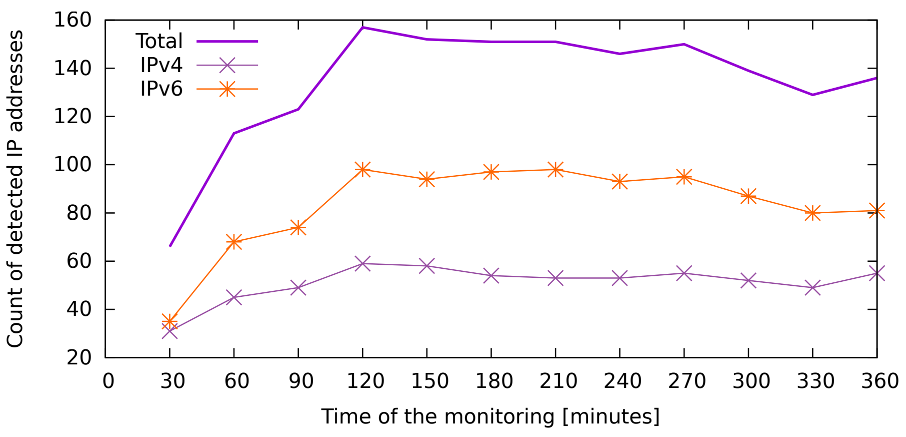


Figure 8.9: The number of IP addresses active in the simulated network.

During the experiment, we compared the identity graph created from information available in local and remote monitoring:

- In local monitoring deployment, the identity graph G_L was created as if it was constructed from RADIUS log files, DHCP log files, and the IPv6 address assignment tracking (as described in Chapter 6). Figure 8.10 shows the average number of linkable IPv6 addresses for each IPv4 address. As the identity graph is constructed from accurate data, the average number of linkable IP addresses shown in Figure 8.10 is the precise number. The linked addresses for each IPv4 address A at time t_e are evaluated by counting IP addresses in the set yielded by $\mathit{linked}(A, \{l_{8.5}, l_{8.13}(t_e, t_e)\}, G_L)$.

- In remote monitoring use case, the identity graph G_R was created as if it was constructed based on clock skew fingerprinting. Let us ignore the clock skew distribution revealed in Chapter 7; suppose a uniform distribution between -60 ppm and 60 ppm. The number of active devices is the same as the number of active IPv4 addresses reported in Figure 8.9. The 120 ppm-wide range is not big enough for so many devices for unique identification with the tolerance of ± 1 ppm. The final identity graph G_R has only one component even though the clock skew estimation was reported with lower uncertainty than ± 1 ppm for long-lasting measurements. Hence, all active IP addresses belong to a single anonymity set when no threshold is applied. Consequently, all active IP addresses are linkable to each other. Note that as Chapter 7 reveals, the real clock skew distribution is in a narrower ppm range.

  When the identity graph uses inaccuracy computed with the custom rules discussed in Appendix B.4, the linkability of the IP addresses can be limited. Figure 8.10 shows the average number of linkable addresses to an IPv4 address when inaccuracy threshold T_i of 3, 5, and 10 is applied by evaluating $\mathit{linked}(A, \{l_{8.5}, l_{8.13}(t_e, t_e), l_{8.19}(T_i)\}, G_R)$ for each IPv4 address A at time t_e and counting IP addresses in the resulting set. As expected, the higher the inaccuracy threshold T_i is, the more IP addresses are in the anonymity set in average. In contrast with the linkability without the threshold, the average number of linked IP addresses is only about 5-times higher than in the local monitoring in the worst case. Consequently, this example shows the benefits of the application of inaccuracy threshold. Although it does not provide precise results for LI, it can help a forensic investigator to limit the number of linked IP addresses.

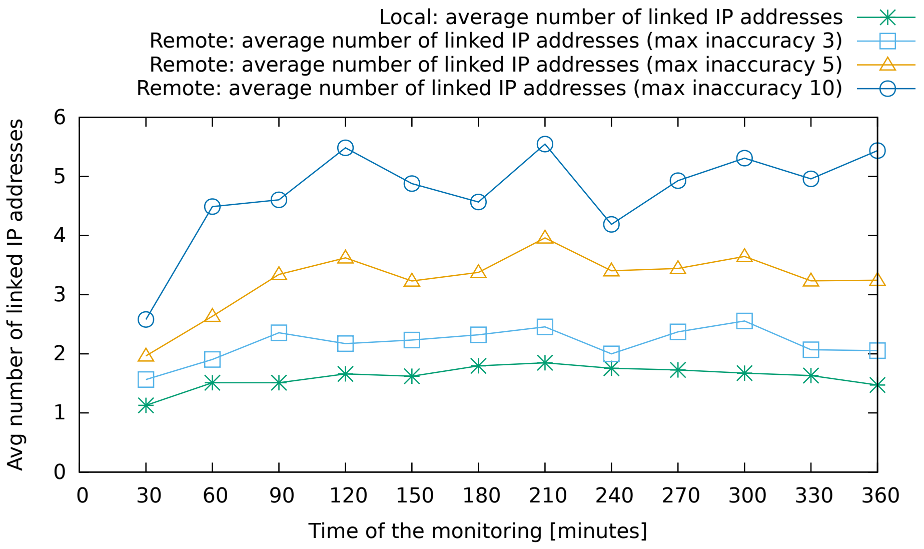


Figure 8.10: The average number of linkable IP addresses to each active IPv4 address during the simulated monitoring.

The example of the simulated network of a small internet service provider also validates the benefits of the time information in identity graphs. It is possible to query information from the past, which is valuable to forensic investigators as explained in Chapter 9.

Appendix B shows additional examples of different networks used for validation of the operations. The examples are also available in the GitHub repository with a testing script[6] that automatically runs all queries and outputs differences to expected results.

8.6 Chapter conclusion

Identity graphs defined in this chapter provide operations that allow an LI system or an investigator to link identifiers based on scope, time, and accuracy. Identity graphs are a framework that allows cross-layer linkability of partial identifiers detected by various partial identity detectors distributed in the network. The operations in identity graph can be extended by defining new constraint function applicable in Formula 8.1 or by extending the set *Categories* or other components of identity graphs.

Identity graphs were tested in SLIS [152, 159] developed under my supervision as a part of the Sec6Net project. Based on the evaluation, this thesis incorporates inaccuracy threshold. In addition, identity graphs defined in this thesis support time-related queries, identifiers of resources (such as a web page URI or a chat room name), and define more operations with the possibility of implementation of custom constraint functions.

Identity graphs support various partial identity detectors including traffic analysers, log analysers, and inaccurate partial identity detectors. During this PhD research, identity graphs were constructed from more than 15 different partial identity detectors [92, 103, 159, 173]. In each deployment, the correct linkage in identity graphs was validated.

[6]`https://github.com/polcak/linking/blob/master/examples/test.sh`

The linking function *linked* defined by Formula 8.1 is generic and supports arbitrary predicates called constraint function to be applied to paths in the graph. This chapter provides examples of constraint function that are useful in the area of LI and further forensic analysis. The generic definition of the function *linked* allows application of constraint function defined in the future.

Identity graphs defined in this chapter validates the Hypothesis 3. Appendix B provides examples of construction of identity graphs and operations in identity graphs. Chapter 9 discusses the applications of the identity graphs including LI, digital forensic, and authentication methods.

Part III

Applicability of the results of this thesis

Chapter 9

Applications

Part II contains the main contribution of this thesis — IPv6 address assignment tracking based on timed transducers in Chapter 6, the research of the clock-skew-based identification in Chapter 7, and identity graphs that link partial identities discovered by various partial identity detectors in Chapter 8. This chapter describes the possible applications of the methods examined in this thesis.

Section 9.1 describes LI systems — the main application of the results of this PhD research. First, Section 9.1 provides general applicability to any LI system. Later, the section describes the application to SLIS [152, 159].

Section 9.2 evaluates the application of this thesis in network forensic, for example, for investigations of traces gathered by LI. First, Section 9.2 provides examples of queries in identity graphs applicable in network forensic; later, the section discusses the application of clock-skew-based-detection.

Section 9.3 evaluates an alternative application of partial identity detectors and identity graphs inside IAN [92, 160]. IAN can be deployed as an enterprise solution that manages network traffic according to the high-level rules defined for specific users or user groups.

Finally, Section 9.4 considers other applications of identity graphs and partial identity detectors — in advertising, web crawling, and customer-centric services.

9.1 Lawful interception systems

This section describes an integration of the partial identity detectors and identity graphs in LI systems with a specific focus on SLIS [159].

Chapter 4 introduces the ETSI LI architecture. Each intercept captures the metadata (IRI), the copy of the communication (CC), or both IRI and CC. Additionally, an LI system has to support intercepts as specified by the warrant and its wording (for example, support for wide range of target identifiers, for interception during the specified period, and for determining the scope of the warrant).

New intercepts can be inserted continuously. Hence, an LI system has to monitor the network state in its internals so that a new interception can start immediately. Consequently, an LI system has to detect all partial identities so that a new interception provides all required data without delay to learn active partial identities in the network. To identify partial identities and to generate the IRI metadata, an LI system has to monitor network traffic, log files, or internals of monitored applications by a partial identity detector as discussed in Section 8.2.

Chapter 6 provides a partial identity detector that tracks IPv6 address assignments to network interfaces identified by a MAC address. Consider the timed transducer $\mathcal{M}$ defined in Section 6.4. The timed transducer monitors an IPv6 address A. The output symbols of $\mathcal{M}$ defined by Equivalence 6.10 contain (1) the information if the IPv6 address A *begins* to be assigned or *ends* to be assigned, and (2) the MAC address of the network interface that assigned the IPv6 address A. The sets of *begin* (see Equivalence 6.8) and *end* (see Equivalence 6.9) symbols can be directly converted into IRI *begin* and IRI *end* records [59] (for more details about the records, see Subsection 4.1.1).

However, an LI system must not generate IRI records for all IPv6 address assignments in the network. Instead, the warrant authorising an intercept allows only IRI records linkable to the target identifier. One option is to modify the deployment of the IPv6 address assignment tracking proposed in Chapter 6:

- If LI captures data based on a set of IPv6 addresses, then, a single timed transducer $\mathcal{M}$ can be deployed to track each IPv6 address in the set.
- If LI captures data based on a set of MAC address S_{MAC}, then the output symbols of $\mathcal{M}$ defined by Equivalencies 6.8 and 6.9 can be limited to the MAC addresses in the set S_{MAC}, that is $V_{\text{MAC}} := S_{\text{MAC}}$. Also, all transitions producing any output need to be updated so that they do not use invalid symbols.

Nevertheless, such modifications has several disadvantages: (1) the input identifiers cannot be linked with other identifiers such as IPv4 addresses, (2) a new interception of an IPv6 address means that a new timed transducer has to be constructed for a previously untracked IPv6 address, and (3) a new interception of a MAC address means that the definition of the address assignment tracking transducer has to be updated to produce *begin* output symbols in case the interface identified by the MAC address has already any IPv6 address assigned.

Identity graphs solve the downsides. An identity graph can be built from the output of the IPv6 address assignment tracking as defined in Section 6.4. Identity graphs reflect changes in partial identities in the network over time continuously. Additionally, *linked* (see Formula 8.1) supports the linking operations for different categories of network identifiers. Consequently, IRI records can be created as described below.

9.1.1 Application of results in Sec6Net Lawful Interception System

SLIS [152, 159] incorporates the IPv6 address assignment tracking and identity graphs. Additionally, *pcf* — the clock-skew-monitoring tool introduced in Chapter 7 can be deployed as another partial identity detector. Moreover, the Sec6Net project provides 11 more partial identity detectors [159]. Identity graphs can be constructed based on information from these partial identity detectors, see Figure 9.1. See Appendix C for examples of SLIS deployments.

SLIS partial identity detectors produce *begin*, *continue*, and *end* messages as described in Chapter 8. Based on these messages, SLIS constructs the identity graph for the monitored network. Additionally, the partial identity detectors provide *report* messages for information that is not a part of a communication session.

For each warrant valid for the network, an authorised personnel can insert intercepts to SLIS, including linking scopes described in Chapter 8, such as *components of a partial identity* or *partial identities of a specific computer*. SLIS automatically monitors the identity graph and active intercepts. For each message from a partial identity detector, SLIS queries

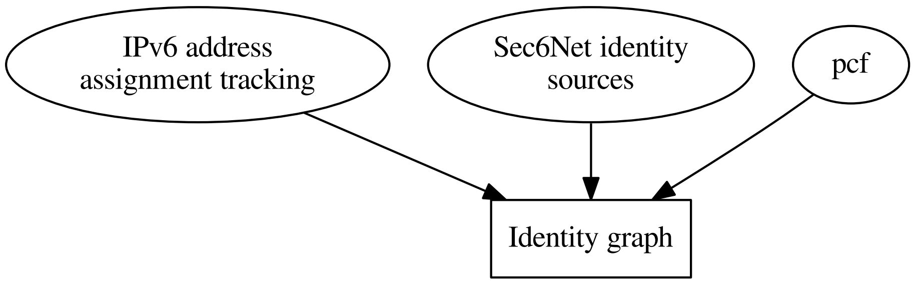


Figure 9.1: A LI system can employ several partial identity detectors to construct the identity graph.

the identity graph for each active intercept if any of the identifiers in the *begin*, *continue*, *end*, or *report* message are linkable to the intercept target identifier. If any identifier is linkable, SLIS creates an IRI record [59] corresponding to the message from the partial identity detector (*begin*, *continue*, *end*, or *report*).

Additionally, for CC interception, SLIS configures hardware accelerated probes developed as a part of the Sec6Net project[1]. The probes filter traffic on wire speed and have to identify the packets to be intercepted quickly. Hence, the probes identify traffic only by category *L4Flow* and *IPAddr* identifiers (see Section 8.3 for more details about the categories). Therefore, SLIS has to employ the constraint function $l_{9.1}$ defined by Formula 9.1 during the linkage by the Formula 8.1 ($l_{9.1} \in L_f$).

$$l_{9.1}(\langle v_0, e_1, v_1, \ldots, e_n, v_n \rangle) \equiv \begin{cases} \text{true} & \text{if } \mathit{category}(v_n) \in \{\mathit{L4Flow}, \mathit{IPAddr}\}, \\ \text{false} & \text{otherwise.} \end{cases} \tag{9.1}$$

Figure 9.2 shows a possible a SLIS deployment in a network of an Internet provider. The identity graph for the network is constructed in the *central device*. Partial identity detectors are deployed in the network, for example, on the link connecting the DHCP server and the RADIUS server to the core network, or on the high-speed application layer probes on the border links. CC-probes intercepts data as close to the revealed intercept target as possible.

The deployment depicted in Figure 9.2 tries to prevent possible confusion and evasion described by Cronin [44]. The confusion si prevented by locating identity detection mechanisms as close to the destination as possible, CC probes are located as close to the intercept target so that all suspect traffic passing the network can be intercepted. A tool *lis-noise-cleaner*[2], developed under my supervision, can detect some confusion attacks [155].

9.1.2 Application of constraint functions in LI

As introduced in the challenge described in Section 5.5, the wording of the warrant can influence the identifiers to be intercepted. Let us list some examples of possible demands requested by warrants, and give linking functions applied in an identity graph G at time t for each demand:

- Intercept the traffic belonging to the user identified by the RADIUS username N:

$$\mathit{linked}(N, \{l_{8.3}, l_{8.13}(t,t), l_{9.1}\}, G) \tag{9.2}$$

[1]Probes for 1 Gbps networks http://www.fit.vutbr.cz/research/prod/index.php?id=428&notitle=1 and 10 Gbps networks http://www.fit.vutbr.cz/research/prod/index.php?id=438&notitle=1

[2]https://www.fit.vutbr.cz/~ipolcak/prods.php?id=307&notitle=1

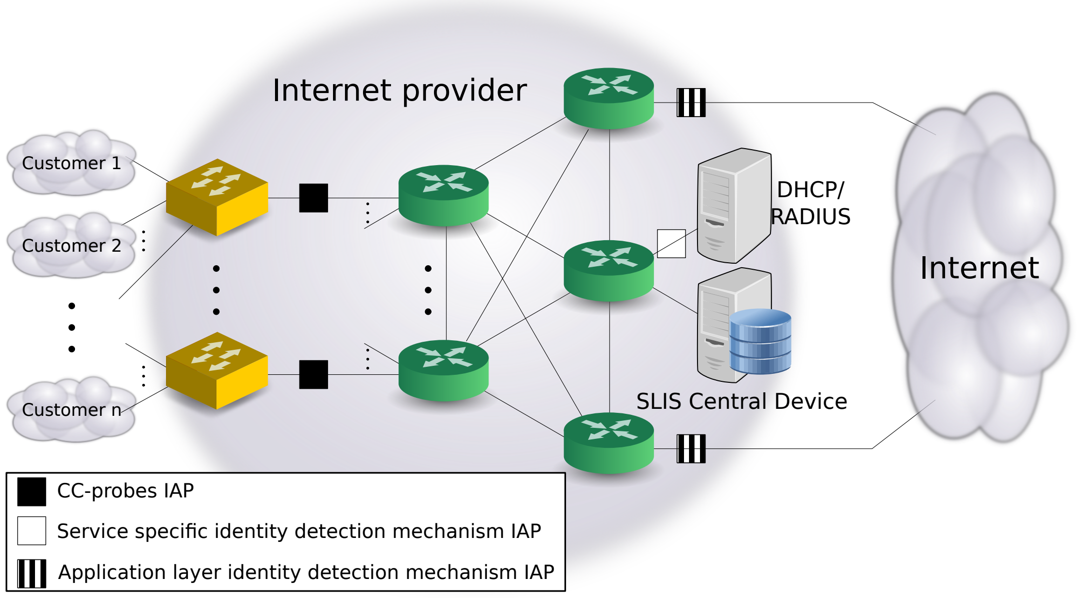

Internet provider
Customer 1
Customer 2
Customer n
DHCP/ RADIUS
SLIS Central Device
Internet
CC-probes IAP
Service specific identity detection mechanism IAP
Application layer identity detection mechanism IAP

Figure 9.2: A possible SLIS deployment.

finds current IP addresses and transport layer flows that are components of the partial identity of N.

- Intercept the traffic belonging to the computer identified by the IPv4 address A:

$$\mathit{linked}(A, \{l_{8.5}, l_{8.13}(t,t), l_{9.1}\}, G) \tag{9.3}$$

finds current IP addresses and transport layer flows that are partial identities of the computer with the IPv4 address A.

- Intercept the traffic of all computers where someone accessed the XMPP account A during last hour:

$$\bigcup_{i \in L} \mathit{linked}(i, \{l_{8.5}, l_{8.13}(t,t), l_{9.1}\}, G), \tag{9.4}$$

where L contains the IP addresses that accessed A during the last hour, that is L contains IP addresses from $L' \equiv \mathit{linked}(A, \{l_{8.7}, l_{8.14}(t-3600, t)\}, G)$.

- Intercept the traffic of all computers where XMPP users that accessed chat room R logged in during the last hour: find all accounts that entered R with the query

$$\mathit{linked}(R, \{l_{8.8}, l_{8.14}(t-3600, t)\}, G) \tag{9.5}$$

and for each account A, intercept the traffic of all computers where someone accessed A as in Formula 9.4.

9.2 Network forensic

This PhD research does not focus on network forensic. However, identity graphs can be a valuable tool for network forensic investigator. This section provides examples of queries that an investigator can execute on identity graphs. Moreover, the forensic investigator can benefit from clock-skew-based identification and estimate clock skew of computers under forensic investigation.

9.2.1 Application of identity graphs in network forensic

Once LEA collects IRI or CC data from an intercept, an investigator can analyse the captured traces with digital forensic tools. This PhD research provides an additional option — the investigator can reconstruct the identity graph G from the captured traces of a single interception or multiple interceptions. Then, the investigator can investigate results of some queries in the identity graphs.

Suppose that the investigator monitors the suspect identified by a RADIUS username R. The investigator can benefit from the results of the following queries:

1. Find all accessed resources from computers authorised by the RADIUS username R by
$$\mathit{linked}(R, \{l_{8.12}\}, G). \tag{9.6}$$
2. For each resource r found by Formula 9.6, find all identifiers of users accessing the resource r by
$$\mathit{linked}(r, \{l_{8.8}\}, G). \tag{9.7}$$
3. For each username n found by Formula 9.7, find all partial identities of computers, from which the accounts were accessed by
$$\mathit{linked}(n, \{l_{8.7}\}, G). \tag{9.8}$$
4. For each computer identifier c found by Formula 9.8, find all usernames accessed from these computers by
$$\mathit{linked}(c, \{l_{8.10}\}, G). \tag{9.9}$$

Additionally, the above queries can be complemented with time-related queries that allow the investigator to reveal additional information such as the usernames with whom the suspect communicate frequently. For example, consider the query

$$\mathit{linked}_{\text{repeatedly}}(R, T, \{l_{8.12}\}, G), \tag{9.10}$$

where T is a set of multiple time periods supplied by the investigator. Such query can limit the number of accessed resources revealed in Formula 9.6 above. Nevertheless, It is up to the investigator to provide the queries that advance the investigation.

9.2.2 Application of clock-skew-based identification in network forensic

A network forensic investigator can estimate clock skew of computers from the traces gathered during an investigation. For example, *pcf* [153] estimates clock skew from pcap files. The possibilities of a passive long-term fingerprinter apply during network forensic investigation as described in Section 7.8.

The long-term duration of clock skew estimates was already applied in the estimation of the geographical location of a tracked computer [130], linking of IP addresses [148] and IPv6 siblings identification [16, 171].

The analysis of real network traces suggests that long-term fingerprinting can reveal similar behaviour on different IP addresses assigned to the same computer even on computers running NTP, see Figure 7.15 for example. In principle, a similar approach can also be applied to transport layer flows to identify flows belonging to the same computer behind NAT.

9.3 High-Level SDN: Identity Aware Networks

A part of this PhD research focused on applications of the method developed during the PhD research outside the area of LI, network forensic, and similar fields. This section introduces an extension of SDN network referred as IAN. In IAN, the knowledge of identities active in the network allows definition and application of specific policies for particular users or groups of users.

Subsection 9.3.1 provides a short introduction to SDN. Subsection 9.3.2 describes high-level SDN controller and examples of IAN applications.

9.3.1 Introduction to SDN

Switches and routers provide a fast hardware fabric that carries data packets to the output port, see Figure 9.3. The fabric is controlled by the content of a table (the CAM table in switches and the routing table in routers) that specifies the output port based on the content specific fields of packet headers. The table is updated by a control process running on the operating system of the device. While the fabric forwards a majority of packets to an output port, some packets, for example, those carrying routing information or unusually formatted packets, are processed in the software and may trigger a change in the CAM or routing table.

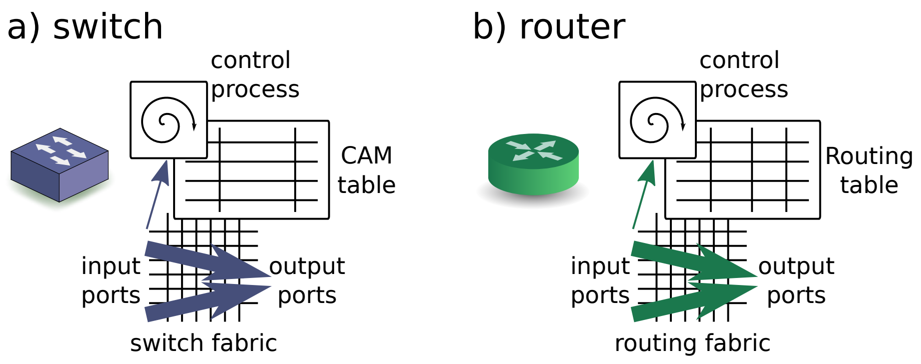


Figure 9.3: Internals of switches and routers.

Typical routers and switches have a fixed set of functions. They support a limited number of protocols. For additional support of a new protocol, the image with the operating system has to be updated. Usually, only the manufacturer of a router or a switch can create a new operating system image. Therefore, routers and switches are inflexible for applications requiring specific behaviour; hence they limit innovation [121].

SDN decouples the data plane (the fabric) and control plane (the control processes). An SDN switch includes a fast data plane fabric and a table that controls the forwarding. The logic that updates the content of the table with rules is offloaded to an SDN controller. An

SDN controller can control a single SDN switch or multiple SDN switches. Several SDN controllers can cooperate and provide backup for each other.

Figure 9.4 shows an example of an SDN network:

- The four SDN switches and links connecting the switches form the data plane. Each switch maintains the connection to at least one SDN controller.
- The control plane contains (1) SDN controllers, (2) links between SDN controllers, (3) links between an SDN controller and SDN switches.

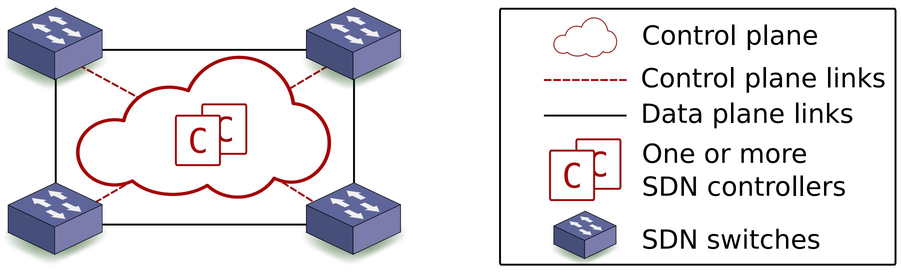


Figure 9.4: SDN switches form a basis of the SDN data plane. Their behavior is orchestrated by one or more SDN controllers. SDN paradigm strongly decouples control and data plane.

Often, an SDN controller provides API for third-party applications and extensions. The APIs allow SDN applications to fine-tune the rules installed to SDN switches. Therefore, it is possible to specify rules suited for needs of a particular network or easily expand the functions of the SDN network. As a result, the SDN paradigm provides tools to extend network functionality.

OpenFlow (OF) [121] is a standardised protocol for communication between an SDN controller and an SDN switch. OF does not differentiate devices according to the ISO/OSI layers [99]. Instead, OF provides rules that are triggered based on input port and header field values of link layer up to transport layer. Each rule can specify all fields or just a subset (non-specified fields are wildcarded). OF supports the following records:

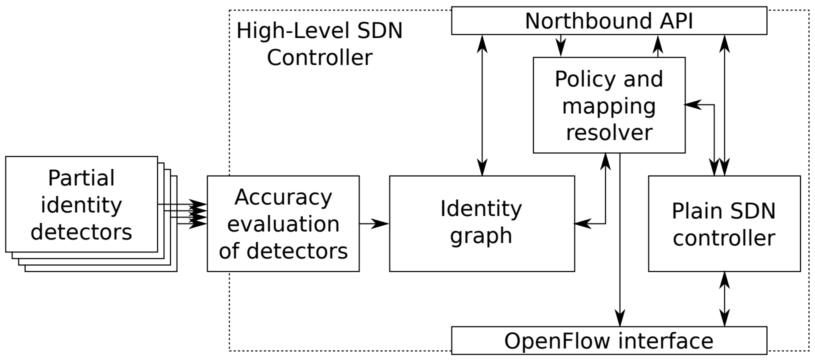
- Identification of the input port.
- Source and destination MAC address, Ethernet Type, VLAN ID.
- MPLS label.
- Source and destination IP address, type of service (flow label), and upper-layer protocol. Both IPv4 and IPv6 are supported.
- Transport layer source and destination port.

A rule can trigger an action, such as packet forwarding, passing the packet to the controller, a change of a header field. Given the generality of rules and actions, OF switch can mimic a switch, a router, a firewall, a load balancer and many other devices or their combinations.

9.3.2 Identities and software defined networking

A part of this PhD research that focused on SDN proposed the idea of high-level policies for SDN [160]. The list of the OF records mentioned above covers items that are network identifiers. However, the records are low-level identifiers that are known for each packet. OF does not allow rules specification for a user or a device. IAN apply high-level SDN policies containing rules identifying computers and users. For example, "Route data of users from department Hotline along path with lowest delay" or "Drop data from users in group Guest to any other user".

Figure 9.5 shows an adapted example of a high-level SDN controller to the terminology used in this thesis. The high-level controller incorporates both partial identity detectors and the identity detector. The detailed description of the internals follows.



High-Level SDN Controller
Northbound API
Policy and mapping resolver
Partial identity detectors
Accuracy evaluation of detectors
Identity graph
Plain SDN controller
OpenFlow interface

Figure 9.5: High-level SDN controller internals — adapted figure [160].

Partial identity detector is defined in Section 8.2.

Accuracy evaluation of detectors assigns or modifies the accuracy advertised by a partial identity detector. See Chapter 8 for details on inaccuracy.

Identity graph is defined in Section 8.4.

Plain SDN controller is any SDN controller (or a hierarchy of SDN controllers) as described in this section.

OpenFlow interface allows OF rules modification in SDN switches.

Northbound API is the *application interface* (API) for custom or third-party SDN or high-level SDN applications. SDN applications communicate with the plain SDN controller, high-level SDN applications communicate with the Policy and mapping resolver. Additionally, high-level SDN applications can query identity graphs and link partial identities. For example, a query is: *What IP addresses are assigned to the devices registered by John Doe at the moment?*

Policy and mapping resolver gathers high-level SDN rules previously inserted by high-level SDN applications. For each rule, the Policy and mapping resolver links the high-level identifiers, such as RADIUS usernames, with identifiers supported by OpenFlow. The queries are evaluated in the identity graph as defined in Section 8.4.

IAN can be deployed in an enterprise network. The partial identity detectors can track, for example, RADIUS, DHCP, DHCPv6, and SLAAC. Based on the relations between identifiers, the identity graph inside the high-level controller can be built. The goal is to learn network identifiers belonging to partial identities of network users. Then, the identity graph links the usernames to the category *IPAddr* and *L4Flow* identifiers. The policy and mapping resolver injects rules containing the identifiers to the SDN switches or the SDN controller. Consequently, network operations can be fine-tuned according to user roles and permissions. For example, it is possible to treat the traffic of the management of a company with higher priority than the traffic of regular employees.

Under my supervision, we considered the following examples of high-level SDN use cases in IAN [92, 160]:

- The quality of Service management — The response time for some traffic, such as the traffic of the management, sales department, hot lines, is more important to the company compared to, for example, accounting. IAN can assign different traffic classes based on the role of the user of the device producing or consuming the traffic.

- Firewall and network access control — Access to some network segments can be restricted to privileged personnel only. Nowadays, it is possible to configure firewalls to limit access to restricted services. However, firewalls typically distinguish network equipment (IP addresses) or applications (for example, by port numbers). Common firewalls do not take into account the identity of the person operating the equipment. Lately, companies allow their employees to bring their own equipment to work (bring your own device policy). IAN can insert dynamic rules that assign privileges depending on the role of the user authenticated on the device.

- Routing — Traffic exchanged through different routes may have a different cost depending on the agreement between the enterprise network and the uplink providers. Hence, some links might be reserved for emergencies. Some user groups can have different emergency thresholds assigned, for example, managers and network operators. IAN can arrange rules that modify routing according to the user identities.

- Network monitoring and accounting — OF provides counters associated with each rule. IAN can aggregate the counter values based on the user identities to provide monitoring and accounting for specific users or groups of users.

- Dual stack captive portals — Subsection 6.1.3 presents a solution for a dual stack captive portal deployed in Kasetsart University network [170]. However, the solution links only a single IPv6 address to a single IPv4 address. IAN can link a single authenticated address to any number of other identifiers revealed by partial identity detectors.

A testbed for IAN was developed and evaluated in a laboratory as a part of this PhD research [92]. The deployment validated the added value of IAN. Nevertheless, IAN were not deployed in a network outside our laboratory.

9.4 Other use cases

This section briefly introduces rough ideas on other use cases that may benefit from the results of this thesis. These use cases were not implemented in practice.

The advertisement industry targets advertisements and provides individual offers according to individual knowledge about each subject, such as the habits, desires, and recent shopping history. The identification of individuals can be divided into the following subtasks:

1. Gathering the attribute values related to each detected partial identity.
2. Linking detected partial identities of the same subject.
3. Selecting an individual offer based on the union of attributes of all partial identities of the subject.

This work does not address the subtask 1 and 3. Nevertheless, identity graphs focus on subtask 2.

In principle, the benefit of applying the results of this work are similar to those offered by Atlas[3] (an advertisement platform operated by Facebook). Atlas offers a possibility to target advertisement to individuals operating several devices. They link browser profiles to a Facebook account. Consequently, an Atlas customer identifies a specific user on all devices (provided that the user has accessed Facebook from the browser profile on the device). This thesis focus on methods that can link partial identities via any identifier. Nevertheless, Facebook has a user base; there is no service maintaining an identity graph of a large user base.

Web crawlers map the web content and index web pages or gather publicly available information from the web, for example, in the context of *Open Source Investigation* (OSINT)[4] [113]. Web crawlers link content created by a specific person. Web crawling can be decoupled to two subtasks:

1. Detection of partial identities that created some content on the web.
2. Linking of detected partial identities, for example, profiles on different web servers.

Identity graphs can be applied to the second subtask.

Customer-centric approach Napatech[5] summarised their thoughts on problems in current networking in a series of white papers. One of the white papers [133] suggests that operators of mobile networks should focus on the customer-centric approach of delivering services. The goal is to improve the satisfaction of the customers. Napatech proposes a distributed solution with deep packet inspection capabilities and application detection. In another white paper of the series [134], Napatech suggests analysis of all network layers. The processing and correlation of the information gathered in the white papers is not clear. Most likely, their solution detects identifiers and attribute values related to a specific traffic. Such solution might benefit from a method to link detected partial identities, such as identity graphs.

[3]http://atlassolutions.com/

[4]The expression *open source* relates to the public availability of the investigation source. It is completely unrelated to *open source software* where *open source* refers to the availability of the source code.

[5]http://www.napatech.com

9.5 Chapter conclusion

This chapter focuses on applications of this PhD research. Primarily, this PhD research focuses on LI. Hence, Section 9.1 presents a complete LI system that controls high-speed LI probes developed at our university.

Besides LI, this chapter considers the deployment of both partial identity detectors and identity graphs in network forensic, high-level SDN, the advertisement industry, web crawlers, and user-centric networks.

The primary goal of this thesis is to provide tools for investigators that identify criminals during LI. As privacy rights prevent surveillance by the general public, there are special requirements for LI codified by law. For other use cases, it is important to stress that both attribute values hoarding and linking of detected partial identities can harm the privacy of individuals covered by the law [39, 180]. Hence, the deployment has to take law and privacy considerations into account.

Chapter 10

Conclusion

This thesis describes identification in modern computer networks compatible with LI. Several challenges arise for LI in modern networks. (1) The shortage of IPv4 addresses caused a massive deployment of NAT that conserve the IPv4 address space. Consequently, several devices can share a single IPv4 address or a pool of IPv4 addresses. (2) Computers regenerate IPv6 addresses frequently. Additionally, SLAAC, the default IPv6 address assignment, is decentralised. Hence, there is not a single point in the network that contains the address assignment information. (3) Currently, IPv6-only deployment is rare. Dual stack networks can multiplex a single session through both IPv4 and IPv6. (4) LI based on application layer identifiers is becoming more significant. (5) Legal requirements mandate that a small change in the wording of a warrant can influence the traffic to be intercepted. Chapter 5 introduces the challenges in detail.

Part II proposes mechanisms that deal with the challenges. IPv6 address tracking defined in Chapter 6 reveals IPv6 addresses in LANs. Chapter 7 evaluates clock-skew-based remote computer identification. Finally, Chapter 8 defines identity graphs that form a framework that supports cross-layer linking of partial identities. The mechanisms were deployed in SLIS, the LI system developed as a part of the Sec6Net project [159]. SLIS configures 1 Gbps and 10 Gbps Sec6Net network probes based on the linking in the identity graph containing identifiers that are active in the network. The operations in identity graphs allow linking of various input identifiers to IP addresses or transport layer flows supported by the probes. SLIS was successfully presented to Czech LEAs.

Besides LI, the knowledge of partial identities can be applied in other use cases. Section 9.2 applies the results in network forensic; Section 9.3 proposes high-level SDN that converts rules for specific users or groups of users to OF rules.

Each chapter in Part II deals with a single hypothesis:

Identification on IPv6 LANs

> **Hypothesis 1**: The detection of IPv6 address assignments is possible from Neighbor Discovery traffic [136] even in networks with MLD snooping enabled (neighbor discovery traffic is multicasted rather than broadcasted).

The IPv6 address assignment tracking based on timed transducers provides a mechanism that detects IPv6 address assignments in networks with MLD snooping (see Chapter 6), which confirms Hypothesis 1. The IPv6 address assignment tracking monitors messages in the network and joins all solicited-node multicast groups in the network. The core of the tracker is formalised as the timed transducer that creates messages that (1) form the basis for IRI records sent to LEMF and (2) enable the construction of identity graphs.

The proposed IPv6 address tracking (1) detects all IPv6 addresses of each node on the LAN, (2) signals newly assigned addresses immediately, (3) does not poll routers in the network, and (4) detects that an IPv6 address was dropped. However, the biggest downside of the proposed IPv6 address assignment tracking is its reliability on the visibility of ND traffic. Each lost message can create inconsistencies between the state of the timed transducer and the network. If desired, the IPv6 address assignment tracking can be further extended to validate the consistency, however, this thesis focused on a method that does not inject additional queries to end hosts and detects the IPv6 address assignments immediately.

Clock-skew-based identification

Hypothesis 2: *Short-term clock skew estimates can uniquely identify computers for LI.*

This thesis also focuses on clock-skew-based identification for a remote computer identification as related research indicates that unique short-term identification is possible [96, 112, 174] and that passive clock skew evaluation is possible [112], see Chapter 7. However, this PhD research has revealed that NTP influences TCP timestamps [158]. The study of real network clock skew distribution shows that most of the real clock skew estimates are close to 0 ppm. Consequently, a short-term fingerprinter cannot reliably identify computers solely by clock skew. Hence, we rejected Hypothesis 2.

Nevertheless, this PhD research provides additional knowledge about the clock-skew-based fingerprinting: (1) the formal requirement for a clock skew measurement, (2) the discovery of the influence of NTP on clock skew, (3) the method that passively links IPv4 and IPv6 addresses of the same computer, (4) the guide that enables a computer to mimic clock skew of another computer, and (5) the study of short-term fingerprinting in a moderately sized dual stack network.

We provide the clock-skew-fingerprinting tool *pcf* [153] that estimates clock skew from live traffic or pcap traces and supports both IPv4 and IPv6. Identity graphs can be built based on the output from *pcf* monitoring live network or pcap traces; Network forensic can benefit from the linkage for further investigation.

Identity graphs

Hypothesis 3: *Identity information revealed at different locations can be linked by applying unambiguous rules.*

Identity graphs store information about network-related identifiers reported by distributed partial identity detectors as defined in Chapter 8, for example, by the proposed IPv6 address tracking and *pcf*. Under my supervision, during the Sec6Net project [159] and as a part of bachelor [22, 91, 103] and diploma [70, 92, 173] theses, more than 15 more partial identity detectors were developed.

Identity graphs support (1) application layer identifiers of persons and resources (such as chat rooms and web pages), (2) an arbitrary number of IPv4 and IPv6 addresses assigned to an interface, and (3) networks with NAT provided that a partial identity detector [80, Chapter 4], [173], reveals information about transport layer flow translation.

Section 8.4 defines identity graphs and operations in identity graphs that can link partial identities with respect to the warrant wording, time, and inaccuracy restrictions. Identity graphs can be extended by additional operations or categories of identifiers. Chapter 9 provides examples of the application of defined operations in LI and network forensic. Appendix B lists examples of operations in identity graphs. Identity graphs validate Hypothesis 3.

10.1 Evaluation of the challenges

Chapter 5 outlined the challenges of identity detection in modern networks. This section shows how the results of this PhD research described in Part II address the challenges.

Network address translation — Identity graphs support networks with NAT in case a partial identity detector reports the related transport layer flows (the category *L4Flow* identifier before the translation and after the translation [80, Chapter 4], [173]). See Chapter 8 and Appendix B for more details.

Addresses in IPv6 Networks — The IPv6 address assignment tracking based on timed transducers defined in Chapter 6 detects IPv6 addresses assigned on LANs manually, by SLAAC, and DHCPv6. The addresses are detected immediately. There is no limit on the number of addresses assigned to an interface. The clock-skew-based remote computer identification links IPv6 addresses of a single computer (see Section 7.5). Identity graphs support both the IPv6 address assignment tracking and *pcf* as partial identity detectors. Identity graphs can link an arbitrary number of IPv6 addresses to a single computer.

Dual-stack networks — The clock-skew-based remote computer identification links an arbitrary number of IPv4 and IPv6 addresses of the same computer (see Section 7.5). Identity graphs support IPv4 addresses and link IPv4 and IPv6 addresses with the possibility of constraint functions that can limit the linkage based on the wording of a warrant. The Sec6Net project developed a DHCP partial identity detector under my supervision [159].

Application layer protocols — Identity graphs support application layer identifiers, namely usernames (category *L7User*) and resources (category *L7Resource*). The operations that link (1) *partial identities of computers where a specific user was authenticated or logged in*, (2) *identifiers of all users accessing a specific resource*, (3) *all user accounts logged in or authenticated from a computer or a set of computers*, and (4) *all accessed resources* defined in Section 8.4 specifically aim on linking of application layer identifiers. Under my supervision, the Sec6Net project developed several partial identities detectors focusing on application layer protocols [22, 159].

Legal requirements — Both IPv6 address assignment tracking and clock-skew-based identification are supposed to be deployed as partial identity detectors that do not generate any traffic that is visible to the end hosts. Identity graphs support operations that link partial identities based on the wording of a warrant. SLIS constructs IRI records from messages received from partial identity detectors based on the queries in the identity graph. SLIS instructs CC-IIF probes to intercept CC data. Hence, the mechanisms described in this thesis tackle the legal challenges specified in Chapter 5.

10.2 Future work

Although this thesis provides answers to hypotheses, new questions arise. This section provides directions for possible future research.

For IPv6 address assignment tracking, the biggest challenge is to remove the dependency on ND traffic. One option is to develop a different method based on an entirely new idea. For example, a router vendor can add an option that automatically signals a new entry in an NC. Another option is to combine ND tracking with another method such as NC polling. The idea is to let time transducers detect address assignments in ND traffic and validate the bindings by NC polling. If a timed transducer misses a new assignment, the NC polling learns the assignment with a delay in case there is some ingress traffic for the IPv6 address.

The IPv4 DHCP logs contain information about leases including the time of the leases and their duration. The access to the DHCP server is typically limited to network operators only. The IPv6 address assignment tracking proposed in Section 6.4 provides similar data to DHCP logs to all locally connected hosts. Hence, from the privacy and security point of view, the proposed IPv6 address assignment tracking leak data that were in IPv4 available to only a privileged group of users. The proposed mechanism works in networks with MLD snooping of the solicited-node multicast groups. The challenge that arises is: can a security mechanism limit DAD so that all local IPv6 addresses do not leak to all local hosts?

For clock-skew-based identification, one of the open problems is the behaviour of mobile devices. Chapter 7 notes that Apple devices exhibit an abnormal clock skew in TCP timestamps. Why? We did not focus on Android as we did not have access to a representative number of devices. However, our experiments show bigger instability of clock skew compared to other Linux distributions.

This PhD research has focused on identification based on passive short-term clock skew estimation. Beverly and Berger [16] combine clock skew fingerprinting and TCP stack fingerprinting. A question for possible future research is if clock skew fingerprinting can be combined with another fingerprinting method so that a passive short-term remote fingerprinting is possible.

Another possible direction in clock-skew-based identification is a specification of an algorithm for matching NTP-influenced clock skew measurements of the same computer. Such algorithm is applicable, for example, in network forensic.

A privacy-related research in clock skew (and hidden identifiers in general) can focus on measures to remove the hidden identifiers. Can the time of the computer be completely decoupled from timestamps sent outside the computer? Can a computer display accurate time to the user and mimic an arbitrary clock skew to the network?

Identity graphs incorporate inaccuracy. However, this thesis does not specify inaccuracy that applies to more inaccurate partial identity detectors simultaneously. Future research can provide a unified methodology that evaluates the inaccuracy of a partial identity detector and inaccuracy of a specific linkage between two partial identities.

Several papers focused on identity linking in the area of social networks [25, 131, 143]. Theoretically, identity graphs should support the crawlers as a partial identity detector. However, we did not construct any identity graph that incorporates data from social networks. Hence, the question is if the current definition of identity graphs is suitable for social network profile linkage.

The FIDIS reported that "virtually any commonly used protocol reveals identifying and linkable information usable for profiling" [65]. The privacy-related question is if a user can prevent such linking.

10.3 Publications related to this thesis

[150] Libor Polčák, Matěj Grégr, Michal Kajan, Petr Matoušek, and Vladimír Veselý. Designing lawful interception in IPv6 networks. In *Security and Protection of Information*, pages 114–126. Brno University of Defence, 2011. ISBN 978-80-7231-777-6

This paper [150] founded initial ideas that ultimately led to Chapters 6 and 8. I was the corresponding author who integrated contributions of other co-authors. I was the main author of Chapters 2 and 3 in the paper. My contribution to Chapters 4 and 6 of the paper is about 50 %.

[155] Libor Polčák, Radek Hranický, and Petr Matoušek. Hiding TCP traffic: Threats and counter-measures. In *Security and Protection of Information 2013, Proceedings of the Conference*, pages 83–96. Brno University of Defence, 2013. ISBN 978-80-7231-922-0

This paper [155] presents the final report for a research of confusion inspired by Cronin et al. [44]. The paper [155] presents a tool to detect confusion in pcap files developed as a part of a bachelor thesis under my supervision [94]. Section 9.1 of this thesis mentions the tool. I am the author of the text of the paper where I adapted the results of the original bachelor thesis [94].

[154] Libor Polčák, Martin Holkovič, and Petr Matoušek. A New Approach for Detection of Host Identity in IPv6 Networks. In *Data Communication Networking*, pages 57–63. SciTePress - Science and Technology Publications, 2013. ISBN 978-989-8565-72-3. Reykjavík, IS

[156] Libor Polčák, Martin Holkovič, and Petr Matoušek. Host Identity Detection in IPv6 Networks. In *Communications in Computer and Information Science*. Springer Berlin Heidelberg, DE, 2014

Both papers [154, 156] provide a basis for Chapter 6 and describe the IPv6 address assignment tracking. The papers do not provide the formal definition of the timed transducer, which is an original contribution of this thesis. I am the author of the text, *ndtrack* was developed as a part of a bachelor thesis under my supervision [91]. The original paper [154] was awarded the Best Student Paper Award at DCNET 2013; I was invited to submit the extended version [156].

[158] Libor Polčák, Jakub Jirásek, and Petr Matoušek. Comments on "Remote physical device fingerprinting". *IEEE Transactions on Dependable and Secure Computing*, 11(5):494–496, 2014. ISSN 1545-5971. Los Alamitos, CA, USA, US

This paper [158] revealed that TCP timestamps are influenced by NTP. I am the author of the text and I searched for the Linux kernel that started to propagate time manipulations to TCP timestamps. Other experiments were adopted from a master thesis that I supervised [104]. This paper initiated the research that ultimately led to the Section 7.4 of this thesis.

[147] Libor Polčák and Barbora Franková. On reliability of clock-skew-based remote computer identification. In *International Conference on Security and Cryptography*. SciTePress - Science and Technology Publications, 2014. Vienna, AT

This paper [147] introduces the idea that the accuracy of a clock skew estimation depends more on the duration of the measurement than the number of samples. Additionally, the paper provides a study of timestamps sources including irregularities. The paper presents a real network clock skew measurement in an IPv4 network. Chapter 7 contains ideas and results from this paper. I am the author of the text in the paper, some of the experiments were adopted from a bachelor thesis [69] that I supervised.

[148] Libor Polčák and Barbora Franková. Clock-skew-based computer identification: Traps and pitfalls. *Journal of Universal Computer Science*, 21(9): 1210–1233, 2015. ISSN 0948-6968

This paper [148] forms a basis for Chapter 7. The paper divides fingerprinters according to the duration and activity of the fingerprinting. The paper provides the (1) formal evaluation of the requirements for precise clock skew estimations (the basis of Section 7.3), (2) the method that links IPv4 and IPv6 addresses of a single computer (the basis of Section 7.5), (3) the attack during which a computer mimics the clock skew of another computer (the basis of Section 7.6), (4) the real network clock skew measurement in the dual stack network (the basis for Section 7.7). I am the author of the text; some experiments were conducted with the help of Barbora Franková under my supervision.

[146] Libor Polčák. Challenges in identification in future computer networks. In *ICETE 2014 Doctoral Consortium*, pages 15–24. SciTePress - Science and Technology Publications, 2014

This paper [146] provides a basic outline of this thesis. The paper won the Best PhD Project Award of the ICETE 2014 Doctoral Consortium.

[157] Libor Polčák, Radek Hranický, and Tomáš Martínek. On identities in modern networks. *The Journal of Digital Forensics, Security and Law*, 2014(2): 9–22, 2014. ISSN 1558-7215

This paper [157] presents the original ideas behind the Chapter 8 and some challenges described in Chapter 5 and describes the application of identity graphs in SLIS (see Section 9.1 for more details). I am the main author of the paper, I wrote the text. Tomáš Martínek contributed with the ideas behind identity graph building. Radek Hranický provided SLIS implementation of identity graphs in his master thesis [95] under my supervision.

[159] Libor Polčák, Tomáš Martínek, Radek Hranický, Stanislav Bárta, Martin Holkovič, Barbora Franková, and Petr Kramoliš. Sec6net: Lawful interception group summary. Technical report, 2014. Faculty of Information Technology Brno University of Technology, FIT-TR-2014-07, Brno, CZ (in Czech)

This technical report [159] concludes the work of the LI group of the Sec6Net project. The technical report describes the partial identity detectors utilised in Chapter 8. This PhD research and the work of the Sec6Net LI group is strongly connected. The technical report contains the application of the ideas presented in Part II of this thesis as described in Section 9.1. I integrated the text of the report based on contributions of all authors.

[160] Libor Polčák, Leo Caldarola, Davide Cuda, Marco Dondero, Domenico Ficara, Barbora Franková, Martin Holkovič, Amine Choukir, Roberto Muccifora, and Antonio Trifilo. High level policies in SDN. In *E-Business and Telecommunications*, volume 2016, pages 39–57. Springer International Publishing, 2016. ISBN 978-3-642-35754-1

This paper [160] focuses on applications of identity graphs and partial identity detectors in SDN (IAN). Section 9.3 is based on the paper. This paper [160] merges IAN and application-aware networks [23] (that I co-developed during my internship at Cisco Systems, Switzerland). I cooperated on IAN with my students in master theses [70, 92] that I supervised. Application-aware networks are a product of Cisco Systems that I joined during the internship.

Bibliography

[1] 6Lab. Web statistics, 2015. Available online at http://6lab.cz/live-statistics/web/, last visit: 2015-08-12.

[2] Limbesh B. Aal, Jignesh N. Parmar, Vishvesh R. Patel, and Dhrubo Jyoti Sen. Whatsapp, Skype, Wickr, Viber, Twitter and Blog are Ready to Asymptote Globally from All Corners during Communications in Latest Fast Life. *Research Journal of Science and Technology*, 6(2):101–116, 2014. ISSN 0975-4393.

[3] Bernard Aboba and Elwyn B. Davies. *Reflections on Internet Transparency*. IETF, 2007. RFC 4924 (Informational).

[4] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. Fpdetective: Dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 1129–1140, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2477-9.

[5] 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, pages 674–689, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2957-6.

[6] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994. ISSN 0304-3975.

[7] AQSACOM. Lawful Interception for IP Network, 2012. White Paper.

[8] Rajiv Asati and Dan Wing. *Tracking of Static/Autoconfigured IPv6 addresses*. Internet Engineering Task Force, 2012. Internet Draft version 00 (Expired Work in progress).

[9] ATIS/TIA. *Lawfully Authorized Electronic Surveillance. J-STD-025-B*. Alliance for Telecommunications Industry Solutions/Telecommunications Industry Association Joint Standard, 2006.

[10] Tuomas Aura. *Cryptographically Generated Addresses (CGA)*. IETF, 2005. RFC 3972 (Proposed Standard).

[11] Tuomas Aura and Alf Zugenmaier. Privacy, control and internet mobility. In *Security Protocols*, Lecture Notes in Computer Science, pages 133–145. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-40925-0. LNCS 3957.

[12] Abdullah Azfar, Kim-Kwang Raymond Choo, and Lin Liu. A study of ten popular android mobile VoIP applications: Are the communications encrypted? In *Proceedings of the 2014 47th Hawaii International Conference on System Sciences*, HICSS '14, pages 4858–4867, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-2504-9.

[13] Fred Baker, Bill Foster, and Chip Sharp. *Cisco Architecture for Lawful Intercept in IP Networks*. IETF, 2004. RFC 3924 (Informational).

[14] Christian Banse, Dominik Herrmann, and Hannes Federrath. Tracking users on the internet with behavioral patterns: Evaluation of its practical feasibility. In *Information Security and Privacy Research*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 235–248. Springer Berlin Heidelberg, DE, 2012. ISBN 978-3-642-30435-4.

[15] Steven M. Bellovin, Matt Blaze, Sandy Clark, and Susan Landau. Lawful hacking: Using existing vulnerabilities for wiretapping on the internet. *Northwestern Journal of Technology and Intellectual Property*, 12:1–65, 2014. ISSN 1549-8271.

[16] Robert Beverly and Arthur Berger. Server siblings: Identifying shared IPv4/IPv6 infrastructure via active fingerprinting. In *Passive and Active Measurement*, Lecture Notes in Computer Science 8995, pages 149–161. Springer International Publishing, 2015. ISBN 978-3-319-15508-1.

[17] Karthikeyan Bhargavan and Carl A. Gunter. Network event recognition. *Formal Methods in System Design*, 27:213–251, 2005. ISSN 0925-9856.

[18] Dave Borman, Bob Braden, Van Jacobson, and Richard Scheffenegger. *TCP Extensions for High Performance*. IETF, 2014. RFC 7323 (Proposed Standard).

[19] Jim Bound, Bernie Volz, Ted Lemon, Charles E. Perkins, and Mike Carney. *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. IETF, 2003. RFC3315 (Proposed Standard).

[20] Robert Braden. *Requirements for Internet Hosts – Communication Layers*, 1989. RFC 1122 (Internet Standard).

[21] Robert Braden. *Requirements for Internet Hosts – Application and Support*, 1989. RFC 1123 (Internet Standard).

[22] Stanislav Bárta. Creating metadata during interception of instant messaging communication, 2012. Bachelor's thesis, Brno University of Technology, CZ (in Czech). Supervisor Libor Polčák.

[23] Leo Caldarola, Amine Choukir, Davide Cuda, Marco Dondero, Domenico Ficara, Roberto Muccifora, Libor Polčák, and Antonio Trifilo. Towards a real application-aware network. In *Proceedings of the 6th International Conference on Data Communication Networking*, pages 5–12. SciTePress - Science and Technology Publications, 2015. ISBN 978-989-758-112-0.

[24] Yinzhi Cao, Song Li, and Erik Wijmans. (Cross-)Browser Fingerprinting via OS and Hardware Level Features. In *Proceedings of Network & Distributed System Security Symposium (NDSS)*, 2017.

[25] Francesca Carmagnola, Francesco Osborne, and Ilaria Torre. User data distributed on the social web: How to identify users on different social systems and collecting data about them. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 9–15, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0407-8.

[26] Eoghan Casey. *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*. Academic Press, Elsevier Inc., USA, 2011. ISBN 978-0-12-374268-1. Third Edition.

[27] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24:84–90, 1981. ISSN 0001-0782.

[28] Kiran K. Chittimaneni, Merike Kaeo, and Eric Vyncke. *Operational Security Considerations for IPv6 Networks*. Internet Engineering Task Force, 2017. Internet Draft version 11 (Work in progress).

[29] Tim Chown, Jari Arkko, Anders Brandt, Ole Troan, and Jason Weil. *IPv6 Home Networking Architecture Principles*. IETF, 2014. RFC 7368 (Informational).

[30] Morten Jagd Christensen, Karen Kimball, and Frank Solensky. *Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches*. IETF, 2006. RFC 4541 (Informational).

[31] Cisco Systems. *Cisco Catalyst 6500 Series Virtual Switching System*, 2012. White Paper.

[32] Cisco Systems. Configure the ASA to pass IPv6 traffic — IPv6 FAQs — Should I use NAT with IPv6?, 2014. URL http://www.cisco.com/c/en/us/support/docs/security/adaptive-security-appliance-asa-software/119012-configure-asa-00.html#anc39.

[33] Cisco Systems. CLI book 2: Cisco ASA series firewall CLI configuration guide, 9.4, 2015. URL http://www.cisco.com/c/en/us/td/docs/security/asa/asa94/config-guides/cli/firewall/asa-94-firewall-config.html.

[34] Sebastian Clauß and Marit Köhntopp. Identity management and its support of multilateral security. *Computer Networks*, 37(2):205–219, 2001. ISSN 1389-1286. Electronic Business Systems.

[35] Lorenzo Colitti, Vint Cerf, Stuart Cheshire, and David Schinazi. *Host address availability recommendations*. IETF, 2016. RFC 7934 (Best Current Practice 204).

[36] *Common Criteria for Information Technology Security Evaluation*. Common Criteria Project, 2012. Version 3.1, Revision 4.

[37] *Communications Assistance for Law enforement Act of 1994 (CALEA)*. Congress of the United States of America.

[38] Congress on Privacy & Surveillance. Congress on Privacy & Surveillance, 2013. Slides and/or transcripts from the talks on the conference are available online at http://ic.epfl.ch/privacy-surveillance, last visit: 2017-05-22.

[39] Constitutional Order of the Czech Republic. Constitutional Act No. 2/1993 Coll. Article 1 – *Listina základních práv a svobod* (Charter of Fundamental Rights and Freedoms).

[40] Alissa Cooper, Fernando Gont, and Dave Thaler. *Security and Privacy Considerations for IPv6 Address Generation Mechanisms.* IETF, 2016. RFC 7721 (Informational).

[41] Council of Europe. Convention on Cybercrime, 2001. ETS No. 185, Budapest Convention.

[42] Matt Crawford. *Transmission of IPv6 Packets over Ethernet Networks.* IETF, 1998. RFC 2464 (Proposed Standard).

[43] Marius Cristea and Bogdan Groza. Fingerprinting smartphones remotely via ICMP timestamps. *IEEE Communications Letters*, 17(6):1081–1083, 2013. ISSN 1089-7798.

[44] Eric Cronin, Micah Sherr, and Matt Blaze. On the (un)reliability of eavesdropping. *International Journal of Security and Networks*, 3:103–113, 2008. ISSN 1747-8405.

[45] Stephen Deering and Robert Hinden. *Internet Protocol, Version 6 (IPv6) Specification.* IETF, 1998. RFC 2460 (Draft Standard).

[46] Stephen E. Deering, William C. Fenner, and Brian Haberman. *Multicast Listener Discovery (MLD) for IPv6*, 1999. RFC 2710 (Proposed Standard).

[47] Alan DeKok. *The Network Access Identifier.* IETF, 2015. RFC 7542 (Proposed Standard).

[48] Amogh Dhamdhere, Matthew Luckie, Bradley Huffaker, kc claffy, Ahmed Elmokashfi, and Emile Aben. Measuring the deployment of IPv6: topology, routing and performance. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, IMC '12, pages 537–550, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1705-4. Boston, Massachusetts, USA.

[49] Ralph Droms. *Dynamic Host Configuration Protocol.* IETF, 1997. RFC 2131 (Draft Standard).

[50] Matthew Dunlop, Stephen Groat, Randy Marchany, and Joseph Tront. The Good, the Bad, the IPv6. In *Communication Networks and Services Research Conference*, pages 77–84, Ottawa, Canada, 2011. ISBN 978-1-4577-0040-8.

[51] Peter Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies*, volume 6205 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, DE, 2010. ISBN 978-3-642-14526-1.

[52] 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, pages 1388–1401, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4139-4.

[53] ETSI. *ETSI TR 101 943: Telecommunications security; Lawful Interception (LI); Concepts of Interception in a generic Network Architecture*. European Telecommunications Standards Institute, 2001. Version 1.1.1.

[54] ETSI. *ETSI TR 101 944: Telecommunications security; Lawful Interception (LI); Issues on IP Interception*. European Telecommunications Standards Institute, 2001. Version 1.1.2.

[55] ETSI. *ETSI ES 201 158: Telecommunications security; Lawful Interception (LI); Requirements for network functions*. European Telecommunications Standards Institute, 2002. Version 1.2.1.

[56] ETSI. *ETSI TR 102 528: Lawful Interception (LI); Interception domain Architecture for IP networks*. European Telecommunications Standards Institute, 2006. Version 1.1.1.

[57] ETSI. *ETSI TR 101 331: Lawful Interception (LI); Requirements of Law Enforcement Agencies*. European Telecommunications Standards Institute, 2009. Version 1.3.1.

[58] ETSI. *ETSI TR 102 232-3: Lawful Interception (LI); Handover Interface and Service-Specific Details (SSD) for IP delivery; Part 3: Service-specific details for internet access services*. European Telecommunications Standards Institute, 2009. Version 2.2.1.

[59] ETSI. *ETSI TR 101 671: Lawful Interception (LI); Handover interface for the lawful interception of telecommunications traffic*. European Telecommunications Standards Institute, 2010. Version 3.6.1.

[60] ETSI. *ETSI TR 102 232-1: Lawful Interception (LI); Handover Interface and Service-Specific Details (SSD) for IP delivery; Part 1: Handover specification for IP delivery*. European Telecommunications Standards Institute, 2010. Version 2.5.1.

[61] ETSI. *ETSI TS 102 232-2: Lawful Interception (LI); Handover Interface and Service-Specific Details (SSD) for IP delivery; Part 2: Service-specific details for E-mail services*. European Telecommunications Standards Institute, 2010. Version 2.5.1.

[62] ETSI. *ETSI TR 102 232-4: Lawful Interception (LI); Handover Interface and Service-Specific Details (SSD) for IP delivery; Part 4: Service-specific details for Layer 2 services*. European Telecommunications Standards Institute, 2010. Version 2.3.1.

[63] ETSI. *ETSI TS 102-232-5: Lawful Interception (LI); Handover Interface and Service-Specific Details (SSD) for IP delivery; Part 5: Service-specific details for IP Multimedia Services*. European Telecommunications Standards Institute, 2010. Version 2.5.1.

[64] Cyrus Farivar. Cisco attributes part of lowered earnings to China's anger toward NSA. Ars Technica, 2013. Available online at http://arstechnica.com/business/2013/11/cisco-attributes-part-of-lowered-earnings-to-chinas-anger-towards-nsa/, last visit: 2015-01-16.

[65] FIDIS project. D3.8: Study on protocols with respect to identity and identification — an insight on network protocols and privacy-aware communication, 2008. Marit Hansen and Ammar Alkassar (ed.), Version 0.8. Available online at http://www.fidis.net/resources/fidis-deliverables/hightechid/#c2216, last visit: 2017-05-22.

[66] Roy T. Fielding and Julian F. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. IETF, 2014. RFC 7231 (Proposed Standard).

[67] Ondřej Filip. Za rok přibylo v ČR více než 100 tisíc uživatelů IPv6, 2013. Blog of CZ.NIC staff. Available online at https://blog.nic.cz/2013/10/23/za-rok-pribylo-v-cr-vice-nez-100-tisic-uzivatelu-ipv6/ (in Czech), last visit: 2015-08-12.

[68] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. *TCP Extensions for Multipath Operation with Multiple Addresses*. IETF, 2013. RFC 6824 (Experimental).

[69] Barbora Franková. Computer identity based on its internal clock skew, 2013. Bachelor's thesis, Brno University of Technology, CZ (in Czech). Supervisor Libor Polčák.

[70] Barbora Franková. Lawful interception in software defined networks, 2015. Master's thesis, Brno University of Technology, CZ (in Czech). Supervisor Libor Polčák.

[71] Sean Gallagher. Googlers say "F*** you" to NSA, company encrypts internal network. Ars Technica, 2013. Available online at http://arstechnica.com/information-technology/2013/11/googlers-say-f-you-to-nsa-company-encrypts-internal-network/, last visit: 2015-01-19.

[72] Rainer Gerhards. *The Syslog Protocol*. IETF, 2009. RFC 5424 (Proposed Standard).

[73] Poonam Ghuli and Rajashree Shettar. A novel approach to implement a shop bot on distributed web crawler. In *IEEE International Advance Computing Conference (IACC)*, pages 882–886, 2014.

[74] Fernando Gont. *A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)*. IETF, 2014. RFC 7217 (Proposed Standard).

[75] Fernando Gont and Tim Chown. *Network Reconnaissance in IPv6 Networks*. IETF, 2016. RFC 7707 (Informational).

[76] Google. Requests for user information — Google Transparency Report, 2016. Available online at https://www.google.com/transparencyreport/userdatarequests/, last visit: 2016-10-30.

[77] Stephen Groat, Matthew Dunlop, Randy Marchany, and Joseph Tront. The privacy implications of stateless IPv6 addressing. In *Cyber Security and Information Intelligence Research*, pages 52:1–52:4, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0017-9. Oak Ridge, Tennessee.

[78] Stephen Groat, Matthew Dunlop, Randy Marchany, and Joseph Tront. What DHCPv6 says about you. In *2011 World Congress on Internet Security*, pages 146–151, London, UK, 2011. ISBN 978-1-4244-8879-7.

[79] Jonathan L. Gross and Jay Yellen. *Graph Theory and Its Applications*. Chapman & Hall/CRC, Taylor & Francis Group, Boca Raton, USA, 2006. ISBN 978-1-58488-505-4. Second Edition.

[80] Matěj Grégr. *User accounting in next generation networks*. PhD thesis, Brno University of Technology, Faculty of Information Technology, 2016. URL http://www.fit.vutbr.cz/study/DP/PD.php?id=448.

[81] Matěj Grégr, Petr Matoušek, Tomáš Podermański, and Miroslav Švéda. Practical IPv6 Monitoring - Challenges and Techniques. In *Symposium on Integrated Network Management*, pages 660–663, Dublin, Ireland, 2011. IEEE CS. ISBN 978-1-4244-9220-6.

[82] Matěj Grégr, Podermański, Tomáš, and Miroslav Švéda. Deploying IPv6 — practical problems from the campus perspective. In *TERENA Networking Conference 2012*, Reykjavik, IS, 2012.

[83] Mark Handley, Vern Paxson, and Christian Kreibich. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *Proceedings of the 10th conference on USENIX Security Symposium - Volume 10*, SSYM'01, 2001.

[84] Hangzhou H3C Technologies. *IRF Stack Technology White Paper*. White Paper, available online at http://www.h3c.com/portal/download.do?id=1890681, last visit: 2015-04-16.

[85] Marit Hansen, Peter Berlich, Jan Camenisch, Sebastian Clauß, Andreas Pfitzmann, and Michael Waidner. Privacy-enhancing identity management. *Information Security Technical Report*, 9(1):35–44, 2004. ISSN 1363-4127.

[86] Uwe Hansmann, Lothar Merk, Martin S. Nicklous, and Thomas Stober. *Pervasive Computing*. Springer-Verlag Berlin Heidelberg, Germany, 2003. ISBN 3-540-00218-9. Second Edition.

[87] Clive Harfield and Karen Harfield. *Covert Investigation*. Oxford University Press Inc, New York, NY, USA, 2008. ISBN 978-0-19-954962-7. Second Edition.

[88] Dominik Herrmann, Karl-Peter Fuchs, and Hannes Federrath. Fingerprinting techniques for target-oriented investigations in network forensics. In *Sicherheit*, volume 7127 of *Lecture Notes in Informatics*, pages 375–390, 2014. ISBN 978-3-88579-622-0.

[89] Robert M. Hinden and Stephen E. Deering. *IP Version 6 Addressing Architecture*. IETF, 2006. RFC 4291 (Draft Standard).

[90] Paul Hoffman and Kornel Terplan. *Intelligence Support Systems: Technologies for Lawful Intercepts*. Auerbach Publications, U.S., 2006. ISBN 978-0-8493-2855-8.

[91] Martin Holkovič. Identity Detection in TCP/IP Architecture, 2013. Bachelor's thesis, Brno University of Technology, CZ (in Slovak). Supervisor Libor Polčák.

[92] Martin Holkovič. SDN Controlled According to User Identity, 2015. Master's thesis, Brno University of Technology, CZ. Supervisor Libor Polčák.

[93] Martin Holkovič and Libor Polčák. Neighbor discovery tracker – ndtrack, 2013. https://www.fit.vutbr.cz/~ipolcak/prods.php?id=308.

[94] Radek Hranický. Fake data in computer networks, 2012. Bachelor's thesis, Brno University of Technology, CZ (in Czech). Supervisor Libor Polčák).

[95] Radek Hranický. Additions to lawful interception system, 2014. Master's thesis, Brno University of Technology, CZ (in Czech). Supervisor Libor Polčák).

[96] Ding-Jie Huang, Kai-Ting Yang, Chien-Chun Ni, Wei-Chung Teng, Tien-Ruey Hsiang, and Yuh-Jye Lee. Clock skew based client device identification in cloud environments. In *Advanced Information Networking and Applications*, pages 526–533, 2012.

[97] IEEE Standards Association. *Guidelines for 64-bit Global Identifier (EUI-64)*, 2012.

[98] *Internet Protocol*. Information Sciences Institute University of Southern California, IETF, 1981. RFC 791 (Internet Standard).

[99] International Organization for Standardization. *ISO/IEC international standard 7498-1:1994 Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*, 1994.

[100] International Organization for Standardization. *ISO/IEC international standard 15408:2009 Information technology – Security techniques – Evaluation criteria for IT security*, 2009.

[101] Van Jacobson, Bob Braden, and Dave Borman. *TCP Extensions for High Performance*. IETF, 1992. RFC 1323 (Proposed Standard, Obsoleted by RFC 7323).

[102] Suman Jana and Sneha K. Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. *IEEE Transactions on Mobile Computing*, 9(3):449–462, 2010. ISSN 1536-1233.

[103] Jakub Jeleň. HTTP-request-based identification, 2015. Bachelor's thesis, Brno University of Technology, CZ (in Slovak). Supervisor Libor Polčák.

[104] Jakub Jirásek. Computer identification using time information, 2012. Master's thesis, Brno University of Technology, CZ (in Czech). Supervisor Libor Polčák.

[105] Audun Jøsang, John Fabre, Brian Hay, James Dalziel, and Simon Pope. Trust requirements in identity management. In *Proceedings of the 2005 Australasian Workshop on Grid Computing and e-Research - Volume 44*, ACSW Frontiers '05, pages 99–108, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc. ISBN 1-920-68226-0. Newcastle, New South Wales, Australia.

[106] Christophe Kalt. *Internet Relay Chat: Architecture*. IETF, 2000. RFC 2810 (Informational).

[107] Christophe Kalt. *Internet Relay Chat: Client Protocol.* IETF, 2000. RFC 2812 (Informational).

[108] Balamurugan Karpagavinayagam, Radu State, and Olivier Festor. Monitoring Architecture for Lawful Interception in VoIP Networks. In *Internet Monitoring and Protection*, 2007.

[109] Juhoon Kim, Nadi Sarrar, and Anja Feldmann. Watching the IPv6 takeoff from an IXP's viewpoint. Technical report, 2014. Forschungsberichte der Fakultät IV Elektrotechnik und Informatik, Technische Universität Berlin, Bericht-Nummer: 2014-01, Berlin, DE.

[110] Matthias Kirchler, Dominik Herrmann, Jens Lindemann, and Marius Kloft. Tracked without a trace: Linking sessions of users by unsupervised learning of patterns in their DNS traffic. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, AISec '16, pages 23–34, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4573-6.

[111] John C. Klensin. *Simple Mail Transfer Protocol.* IETF, 2008. RFC 5321 (Draft Standard).

[112] Tadayoshi Kohno, Andre Broido, and Kimberly C. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2): 93–108, 2005. ISSN 1545-5971.

[113] Bert-Jaap Koops. Police investigations in Internet open sources: Procedural-law issues. *Computer Law & Security Review*, 29(6):654–665, 2013. ISSN 0267-3649.

[114] Julius Kriukas. addrwatch: A tool similar to arpwatch for IPv4/IPv6 and ethernet address pairing monitoring, 2012. https://github.com/fln/addrwatch.

[115] Susan Landau. *Surveillance or security? : the risks posed by new wiretapping technologies.* The MIT Press, Cambridge, MA, USA, 2010. ISBN 978-0-262-01530-1.

[116] Fabian Lanze, Andriy Panchenko, Benjamin Braatz, and Andreas Zinnen. Clock skew based remote device fingerprinting demystified. In *Global Communications Conference*, pages 813–819, 2012.

[117] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 878–894, 2016.

[118] Law Order of the Czech Republic. Act No. 127/2005 Coll. and its subsequent amendments and additions *Zákon o elektronických komunikacích* accompanied by the Decree No. 336/2005 Coll. (In Czech).

[119] Steven Levy. How the NSA Almost Killed the Internet. Wired.com, 2014. Available online at `http://www.wired.com/2014/01/how-the-us-almost-killed-the-internet/all/`, last visit: 2015-01-19.

[120] Louis Mamakos, Kurt Lidl, Jeff Evarts, David Carrel, Dan Simone, and Ross Wheeler. *A Method for Transmitting PPP Over Ethernet (PPPoE).* IETF, 1999. RFC 2516 (Informational).

[121] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Computer Communication Review*, 38 (2):69–74, 2008. ISSN 0146-4833.

[122] David Medine, Rachel Brand, Elisebeth Collins Cook, James Dempsey, and Patricia Wald. Recommendations Assessment Report, 2015. Privacy and Civil Liberties Oversight Board. Available online at http://www.pclob.gov/newsroom/20150129.html, last visit 2015-02-10.

[123] Martin Meints and Mark Gasson. High-tech id and emerging technologies. In Kai Rannenberg, Denis Royer, and André Deuker, editors, *The Future of Identity in the Information Society*, pages 130–189. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-88480-4.

[124] Andro Milanović, Siniša Srbljić, Ivo Ražnjević, Darryl Sladden, Ivan Matošević, and Daniel Skrobo. Methods for lawful interception in IP telephony networks based on H.323. In *EUROCON 2003. Computer as a Tool.*, volume 1, pages 198–202, 2003.

[125] Andro Milanović, Siniša Srbljić, Ivo Ražnjević, Darryl Sladden, Daniel Skrobo, and Ivan Matošević. Distributed system for lawful interception in VoIP networks. In *EUROCON 2003. Computer as a Tool.*, volume 1, pages 203–207, 2003.

[126] David L. Mills, Jim Martin, Jack Burbank, and William Kasch. *Network Time Protocol Version 4: Protocol and Algorithms Specification*. IETF, 2010. RFC 5905 (Proposed Standard).

[127] Nick 'Sharkey' Moore. *Optimistic Duplicate Address Detection (DAD) for IPv6*. IETF, 2006. RFC 4429 (Proposed Standard).

[128] Keaton Mowery and Hovav Shacham. Pixel Perfect: Fingerprinting Canvas in HTML5. In *Proceedings of W2SP*, 2012.

[129] Robert Mueller. Testimony before U.S. Congress, House of Representatives, Comittee on the Judiciary, Federal Bureau of Investigation, Part II, One Hundred and Tenth Congress, Second Session, 2008. Serial No. 110-99, Available online at http://www.fbi.gov/news/testimony/fbi-priorities-changes-and-challenges-3, last visit: 2015-02-06.

[130] Steven J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Computer and Communications Security*, pages 27–36, New York, NY, USA, 2006. ACM. ISBN 1-59593-518-5. Alexandria, Virginia, USA.

[131] Y. Na, Z. Yinliang, D. Lili, B. Genqing, E. Liu, and G. J. Clapworthy. User identification based on multiple attribute decision making in social networks. *China Communications*, 10(12):37–49, 2013. ISSN 1673-5447.

[132] Tierry Nabeth. Identity of identity. In Kai Rannenberg, Denis Royer, and André Deuker, editors, *The Future of Identity in the Information Society*, pages 19–70. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-88480-4.

[133] Napatech. Time to ReThink Mobile Network Analysis, 2014. White paper, version 6, available online at http://www.napatech.com/sites/default/files/dn-0720_ttrt_mobile_network_analysis_v06_us_a4_online.pdf.

[134] Napatech. Time to ReThink Performance Monitoring, 2014. White paper, version 6, available online at http://www.napatech.com/sites/default/files/dn-0645_ttrt_performance_monitoring_v6_us_a4_online.pdf.

[135] Thomas Narten, Richard Draves, and Suresh Krishnan. *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. IETF, 2007. RFC 4941 (Draft Standard).

[136] Thomas Narten, Erik Nordmark, William Allen Simpson, and Hesham Soliman. *Neighbor Discovery for IP version 6 (IPv6)*. IETF, 2007. RFC 4861 (Draft Standard).

[137] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *2013 IEEE Symposium on Security and Privacy*, pages 541–555, 2013.

[138] Office of the Director of National Intelligence. Signals Intelligence Reform — 2015 Anniversary Report, 2015. Available online at http://icontherecord.tumblr.com/ppd-28/2015/overview, last visit 2015-02-10.

[139] Komang Oka Saputra, Wei-Chung Teng, and Takaaki Nara. Hough transform-based clock skew measurement over network. *IEICE Transactions on Information and Systems*, E99(8):2100 – 2108, 2015. ISSN 0916-8532.

[140] Konstantina Papagiannaki, Sue Moon, Chuck Fraleigh, Patrick Thiran, Fouad Tobagi, and Christophe Diot. Analysis of measured single-hop delay from an operational backbone network. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 535–544, 2002.

[141] Ronald Pashby. *Simplifying IPv6 MLD Snooping Switches*. Internet Engineering Task Force, 2006. Internet Draft version 01 (Expired Work in progress).

[142] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435 – 2463, 1999.

[143] Olga Peled, Michael Fire, Lior Rokach, and Yuval Elovici. Entity matching in online social networks. In *2013 International Conference on Social Computing*, pages 339–344, 2013.

[144] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. Technical report, 2010. Version 0.34, Available online at https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf.

[145] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudeonymity — a proposal for terminology. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pages 1–9, New York, NY, USA, 2001. Springer-Verlag New York, Inc. ISBN 3-540-41724-9. Berkeley, California, USA.

[146] Libor Polčák. Challenges in identification in future computer networks. In *ICETE 2014 Doctoral Consortium*, pages 15–24. SciTePress - Science and Technology Publications, 2014.

[147] Libor Polčák and Barbora Franková. On reliability of clock-skew-based remote computer identification. In *International Conference on Security and Cryptography*. SciTePress - Science and Technology Publications, 2014. Vienna, AT.

[148] Libor Polčák and Barbora Franková. Clock-skew-based computer identification: Traps and pitfalls. *Journal of Universal Computer Science*, 21(9):1210–1233, 2015. ISSN 0948-6968.

[149] Libor Polčák and Martin Holkovič. Behaviour of various operating systems during SLAAC, DAD, and ND, 2013. http://6lab.cz/?p=1691.

[150] Libor Polčák, Matěj Grégr, Michal Kajan, Petr Matoušek, and Vladimír Veselý. Designing lawful interception in IPv6 networks. In *Security and Protection of Information*, pages 114–126. Brno University of Defence, 2011. ISBN 978-80-7231-777-6.

[151] Libor Polčák, Tomáš Martínek, Radek Hranický, Stanislav Bárta, Martin Holkovič, Barbora Franková, and Petr Kramoliš. Sec6Net Identity Management System, 2011–2014. https://www.fit.vutbr.cz/~ipolcak/prods.php.en?id=399&notitle=1.

[152] Libor Polčák, Tomáš Martínek, Radek Hranický, Stanislav Bárta, Martin Holkovič, Barbora Franková, and Petr Kramoliš. Sec6Net Lawful Interception System, 2011-2014. https://www.fit.vutbr.cz/~ipolcak/prods.php.en?id=397&notitle=1.

[153] Libor Polčák, Jakub Jirásek, and Barbora Franková. PC Fingerprinter – pcf, 2012-2014. https://github.com/polcak/pcf.

[154] Libor Polčák, Martin Holkovič, and Petr Matoušek. A New Approach for Detection of Host Identity in IPv6 Networks. In *Data Communication Networking*, pages 57–63. SciTePress - Science and Technology Publications, 2013. ISBN 978-989-8565-72-3. Reykjavík, IS.

[155] Libor Polčák, Radek Hranický, and Petr Matoušek. Hiding TCP traffic: Threats and counter-measures. In *Security and Protection of Information 2013, Proceedings of the Conference*, pages 83–96. Brno University of Defence, 2013. ISBN 978-80-7231-922-0.

[156] Libor Polčák, Martin Holkovič, and Petr Matoušek. Host Identity Detection in IPv6 Networks. In *Communications in Computer and Information Science*. Springer Berlin Heidelberg, DE, 2014.

[157] Libor Polčák, Radek Hranický, and Tomáš Martínek. On identities in modern networks. *The Journal of Digital Forensics, Security and Law*, 2014(2):9–22, 2014. ISSN 1558-7215.

[158] Libor Polčák, Jakub Jirásek, and Petr Matoušek. Comments on "Remote physical device fingerprinting". *IEEE Transactions on Dependable and Secure Computing*, 11 (5):494–496, 2014. ISSN 1545-5971. Los Alamitos, CA, USA, US.

[159] Libor Polčák, Tomáš Martínek, Radek Hranický, Stanislav Bárta, Martin Holkovič, Barbora Franková, and Petr Kramoliš. Sec6net: Lawful interception group summary. Technical report, 2014. Faculty of Information Technology Brno University of Technology, FIT-TR-2014-07, Brno, CZ (in Czech).

[160] Libor Polčák, Leo Caldarola, Davide Cuda, Marco Dondero, Domenico Ficara, Barbora Franková, Martin Holkovič, Amine Choukir, Roberto Muccifora, and Antonio Trifilo. High level policies in SDN. In *E-Business and Telecommunications*, volume 2016, pages 39–57. Springer International Publishing, 2016. ISBN 978-3-642-35754-1.

[161] John Postel. *Internet Protocol*, 1981. RFC 792 (Internet Standard).

[162] Kai Rannenberg, Denis Royer, and André Deuker. Introduction. In *The Future of Identity in the Information Society*, pages 1–11. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-88480-4.

[163] Yakov Rekhter, Robert G Moskowitz, Daniel Karrenberg, Geert Jan de Groot, and Eliot Lear. *Address Allocation for Private Internets*. IETF, 1996. RFC 1918 (Best Current Practice 5).

[164] Peter Resnick. *Internet Message Format*. IETF, 2008. RFC 5322 (Draft Standard).

[165] Carl Rigney, Allan C. Rubens, William Allen Simpson, and Steve Willens. *Remote Authentication Dial In User Service (RADIUS)*. IETF, 2000. RFC 2865 (Draft Standard).

[166] Andres Rojas, Philip Branch, and Grenville Armitage. Predictive lawful interception in mobile ipv6 networks. In *15th IEEE International Conference on Networks*, pages 501–506, 2007.

[167] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. *SIP: Session Initiation Protocol*. IETF, 2002. RFC3261 (Proposed Standard).

[168] Réseaux IP Européens Network Coordination Centre (RIPE NCC). IPv4 Exhaustion, 2012. Available online at `http://www.ripe.net/internet-coordination/ipv4-exhaustion`, last visit: 2015-04-10.

[169] Peter Saint-Andre. *Extensible Messaging and Presence Protocol (XMPP): Core*. IETF, 2011. RFC 6120 (Proposed Standard).

[170] Surasak Sanguanpong and Kasom Koht-Arsa. A design and implementation of dual-stack aware authentication system for enterprise captive portal. In *9th International Conference on Network and Service Management (CNSM)*, pages 118–121, Zürich, Switzerland, 2013. ISBN 978-3-901882-53-1.

[171] Quirin Scheitle, Oliver Gasser, Minoo Rouhi, and Georg Carle. Large-scale classification of ipv4-ipv6 siblings with nonlinear clock skew. *Computing Research Repository*, abs/1610.07251, 2016. URL http://arxiv.org/abs/1610.07251.

[172] Sebastian Schrittwieser, Peter Frühwirt, Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Markus Huber, and Edgar Weippl. Guess who's texting you? evaluating the security of smartphone messaging applications. In *Proceedings of the Network and Distributed System Security Symposium*. The Internet Society, 2012.

[173] Michal Šeptun. Identities in tunelled networks and during network address translation, 2015. Master's thesis, Brno University of Technology, CZ (in Czech). Supervisor Libor Polčák.

[174] Swati Sharma, Alefiya Hussain, and Huzur Saran. Experience with heterogenous clock-skew based device fingerprinting. In *Workshop on Learning from Authoritative Security Experiment Results*, pages 9–18. ACM, 2012. ISBN 978-1-4503-1195-3. Arlington, Virginia.

[175] Vladislav Shkapenyuk and Torsten Suel. Design and implementation of a high-performance distributed web crawler. In *18th International Conference on Data Engineering, 2002. Proceedings*, pages 357–368, 2002.

[176] Modinis IDM Study Team. Modinis study on identity management in egovernment. Technical report, 2005. URL http://ec.europa.eu/information_society/activities/ict_psp/documents/eidm_conceptual_framework.pdf. Service Contract number 29042, prepared for European Commission, Information Society and Media Directorate-General, eGovernment Unit.

[177] TeliaSonera AB. TeliaSonera Sustainability Report 2013, 2014. Available online at http://annualreports.teliasonera.com/global/2013/sustainability%20report/downloads_finala%20pdf/teliasonera_sr2013.pdf, last visit: 2015-01-19.

[178] Dave Thaler, Lixia Zhang, and Gregory Lebovitz. *IAB Thoughts on IPv6 Network Address Translation*. IETF, 2010. RFC 5902 (Informational).

[179] The Council of the European Union. COUNCIL RESOLUTION of 17 January 1995 on the lawful interception of telecommunications (96/C 329/01), 1996.

[180] The European Parliament and The Council of the European Union. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data , 1995.

[181] The Institute of Electrical and Electronics Engineers, Inc. *802®: IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*. 2014. ISBN 978-0-7381-9219-2.

[182] The Internet Assigned Numbers Authority. IANA IPv4 address space registry, 2014. Available online at http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml, last visit: 2015-04-10.

[183] The White House — Office of the Press Secretary. Statement by Assistant to the President for Homeland Security and Counterterrorism Lisa Monaco: Update on Implementation of Signals Intelligence Reform and Issuance of PPD-28, 2015. Available online at http://www.whitehouse.gov/the-press-office/2015/02/03/statement-assistant-president-homeland-security-and-counterterrorism-lis, last visit 2015-02-10.

[184] Susan Thomson, Thomas Narten, and Tatuya Jinmei. *IPv6 Stateless Address Autoconfiguration*. IETF, 2007. RFC 4862 (Draft Standard).

[185] Jenny Torres, Michele Nogueira, and Guy Pujolle. A survey on identity management for the future network. *IEEE Communications Surveys & Tutorials*, 15 (2):787–802, 2013. ISSN 1553-877X.

[186] Ole Troan and Ralph Droms. *IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6*. IETF, 2003. RFC3315 (Proposed Standard).

[187] Utimaco TS GmbH. Lawful interception in the digital age: Vital elements of an effective solution, 2014. White Paper.

[188] Vodafone Group Plc. Law Enforcement Disclosure Report, 2014. Available online at http://www.vodafone.com/content/sustainabilityreport/2014/index/operating_responsibly/privacy_and_security/law_enforcement.html, last visit: 2015-01-16.

[189] Vodafone Group Plc. Law Enforcement Disclosure Report: Legal Annexe, 2014. Available online at http://www.vodafone.com/content/dam/sustainability/2014/pdf/operating-responsibly/vodafone_law_enforcement_disclosure_report.pdf, last visit: 2015-01-16.

[190] Vodafone Group Plc. Vodafone Group Plc Sustainability Report 2013/14, 2014. Available online at http://www.vodafone.com/content/dam/sustainability/2014/pdf/vodafone_full_report_2014.pdf, last visit: 2015-01-16.

[191] Eric Vyncke, Pascal Thubert, Eric Levy-Abegnoli, and Andrew Yourtchenko. *Why Network-Layer Multicast is Not Always Efficient At Datalink Layer*. Internet Engineering Task Force, 2014. Internet Draft version 01 (Expired Work in progress).

[192] Margaret Wasserman and Fred Baker. *IPv6-to-IPv6 Network Prefix Translation*. IETF, 2011. RFC 6296 (Experimental).

[193] Jason Weil, Victor Kuarsingh, Chris Donley, Christopher Liljenstolpe, and Marla Azinger. *IANA-Reserved IPv4 Prefix for Shared Address Space*. IETF, 2012. RFC 6598 (Best Current Practice 153).

[194] Dan Wing and Andrew Yourtchenko. *Happy Eyeballs: Success with Dual-Stack Hosts*. IETF, 2012. RFC 6555 (Proposed Standard).

[195] Menghui Yang and Hua Liu. Implementation and performance of VoIP interception based on SIP session border controller. *Telecommunication Systems*, 55(3):345–361, 2014. ISSN 1018-4864.

[196] Andrew Yourtchenko and Lorenzo Colitti. *Reducing energy consumption of Router Advertisements.* IETF, 2016. RFC 7772 (Best Current Practice 202).

[197] Andrew Yourtchenko and Erik Nordmark. *A survey of issues related to IPv6 Duplicate Address Detection.* Internet Engineering Task Force, 2015. Internet Draft version 01 (Expired Work in progress).

[198] Sebastian Zander and Steven J. Murdoch. An improved clock-skew measurement technique for revealing hidden services. In *Proceedings of the 17th Conference on Security Symposium*, pages 211–225, Berkeley, CA, USA, 2008. USENIX Association.

Part IV

Appendices

Appendix A

Proposed clock skew estimation

Algorithm 7.3 defines clock skew estimation as implemented in *pcf*. This appendix provides an example of the clock skew estimation.

(1) Algorithm 7.3, step 1: *Previous batch* is empty.

(2) Algorithm 7.3, step 2: Collect a batch of timestamps. Figure A.1 shows collected offset points for the first batch of timestamps.

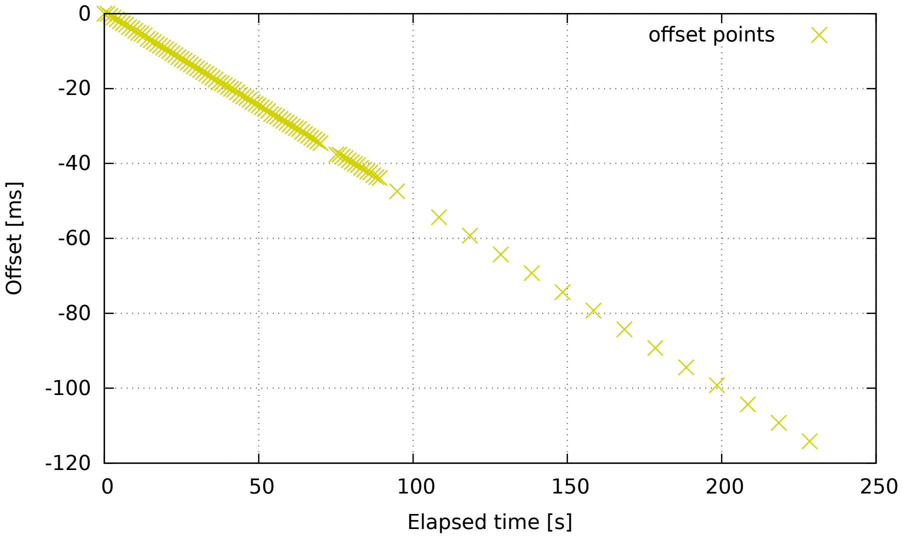


Figure A.1: First batch of timestamps received and offset points computed.

(3) Algorithm 7.3, step 3: Estimate the clock skew based on offset points generated for the timestamps in the *current batch*. *Previous batch* is empty, so set *current batch* as *previous batch* and go to step 2 of Algorithm 7.3.

(4) Algorithm 7.3, step 2: Collect a batch of timestamps.

(5) Algorithm 7.3, step 3: Estimate the clock skew based on offset points generated for the timestamps in the *current batch*.

(6) Algorithm 7.3, step 4: Compare the estimate for *current batch* with the previous estimate. Suppose that the estimated clock skew for this batch is similar to the clock skew estimate computed in step (3).

(7) Algorithm 7.3, step 5: Append offset points in *current batch* to the *previous batch* and estimate clock skew based on all merged offset points. Figure A.2 shows collected

offset points of both batches and the estimated clock skew. So far the observed clock skew is stable.

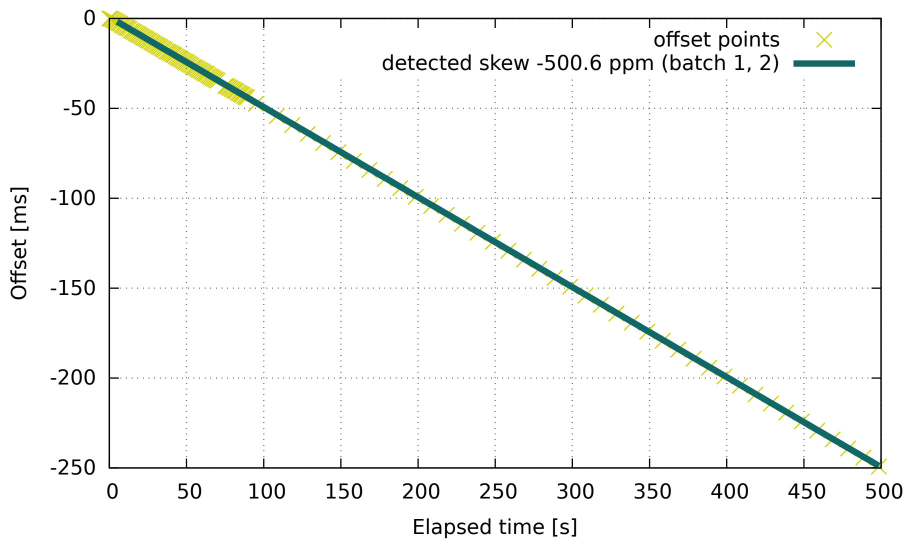


Figure A.2: Clockskew estimated for the first two batches of timestamps.

(8) Algorithm 7.3, step 2: Collect a batch of timestamps.

(9) Algorithm 7.3, step 3: Estimate the clock skew based on offset points generated for the timestamps in the *current batch*. Figure A.3 depicts the estimation.

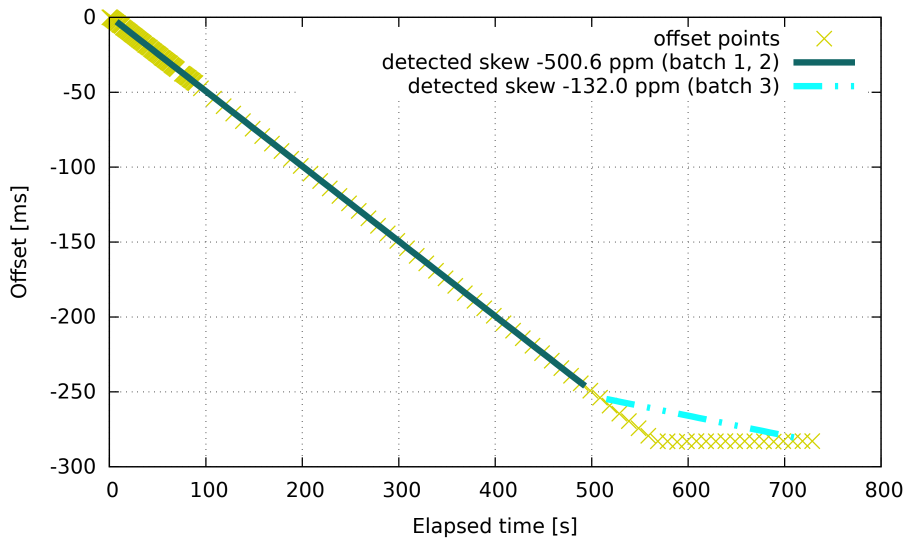


Figure A.3: Clock skew estimated for the third batch differs by 368.6 ppm.

(10) Algorithm 7.3, step 4: Compare the estimate -132.0 ppm to -500.6 ppm. As the difference of 368.6 ppm is bigger than 10 ppm, clock skew changed during the period in which the third batch of timestamps arrived.

(11) Algorithm 7.3, step 6: Ignore *current batch* as the observed clock skew changed in the *current batch* of timestamps.

(12) Algorithm 7.3, step 1: Set *previous batch* to empty.

(13) Algorithm 7.3, step 2: Collect a batch of timestamps.

(14) Algorithm 7.3, step 3: Estimate the clock skew based on offset points generated for the timestamps in the *current batch*. Figure A.4 shows the computed offset points and detected segments of stable clock skew. *Previous batch* is empty, so set *current batch* as *previous batch* and go to step 2 of Algorithm 7.3.

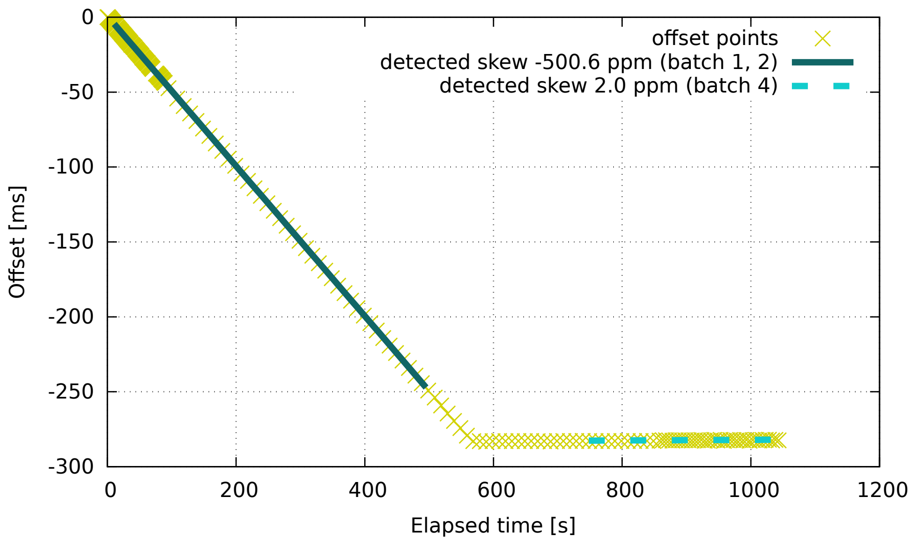


Figure A.4: Clock skew estimated for the fourth batch is stable.

(15) Continue with the clock skew monitoring.

Appendix B

Examples of identity graphs and operations

Chapter 8 defines identity graphs, *linked* (see Formula 8.1), and several constraint functions. This appendix provides examples of their applications.

B.1 Construction of identity graph

This section focuses on building identity graphs based on Algorithms 8.1, 8.2, and 8.3.

In the initial state, there is no knowledge of the network state and the initial identity graph $G = (V, E, \mathit{endpts}, \mathit{category}, \mathit{attributes})$ is empty:

- $V := \emptyset$,
- $E := \emptyset$,
- The domains of *endpts*, *category*, and *attributes* are empty.

DHCP partial identity detector detects a new lease — the DHCP partial identity detector sends the following *begin* message m_1:

- timestamp: 1494320000,
- identifiers: IPv4 address *10.0.0.1*, MAC address *00:11:22:33:44:55*,

According to Algorithm 8.1, the components of G are redefined:

- $V := \{\text{IPv4: } 10.0.0.1, \text{MAC: } 00{:}11{:}22{:}33{:}44{:}55\}$,
- $E := \{e_1\}$,
- $\mathit{endpts} := \{(e_1, \{\text{IPv4: } 10.0.0.1, \text{MAC: } 00{:}11{:}22{:}33{:}44{:}55\})\}$,
- $\mathit{category} := \{(\text{IPv4: } 10.0.0.1, \mathit{IPAddr}), (\text{MAC: } 00{:}11{:}22{:}33{:}44{:}55, \mathit{IfcOrComp})\}$,
- $\mathit{attributes} := \{(e_1, (\text{DHCP}, 1494320000, \infty, 0))\}$.
- Additionally, remember that e_1 was added by the DHCP partial identity detector for the *begin* message m_1.

DHCP partial identity detector detects a renewal for the lease — the DHCP partial identity detector sends the following *continue* message m_2:

- timestamp: 1494320300,
- identifiers: IPv4 address *10.0.0.1*, MAC address *00:11:22:33:44:55*,
- inaccuracy: 0 ($i := 0$).

According to Algorithm 8.2, as $\Pi_4(\mathit{attributes}(e_1)) = i$, nothing changes in the components of G.

Clock skew partial identity detector estimates clock skew of the computer — the clock skew partial identity detector fingerprints the computer with IPv4 address *10.0.0.1* and sends the following *begin* message m_3:

- timestamp: 1494320302,
- identifiers: IPv4 address *10.0.0.1*, clock skew *20 ppm*,
- inaccuracy: 2.

According to Algorithm 8.1, the components of G are redefined:

- $V := \{\text{IPv4: }10.0.0.1, \text{MAC: }00{:}11{:}22{:}33{:}44{:}55, \text{skew }20\,\text{ppm}\}$,
- $E := \{e_1, e_2\}$,
- $\mathit{endpts} := \{(e_1, \{\text{IPv4: }10.0.0.1, \text{MAC: }00{:}11{:}22{:}33{:}44{:}55\}),$
  $(e_2, \{\text{IPv4: }10.0.0.1, \text{skew }20\,\text{ppm}\})\}$,
- $\mathit{category} := \{(\text{IPv4: }10.0.0.1, \mathit{IPAddr}), (\text{MAC: }00{:}11{:}22{:}33{:}44{:}55, \mathit{IfcOrComp}),$
  $(\text{skew }20\,\text{ppm}, \mathit{IfcOrComp})\}$,
- $\mathit{attributes} := \{(e_1, (\text{DHCP}, 1494320000, \infty, 0)), (e_2, (\text{clockskew}, 1494320302, \infty, 2))\}$.
- Additionally, remember that e_2 was added by the clock skew partial identity detector for the *begin* message m_3.

Clock skew partial identity detector improves the clock skew estimation of the computer — the clock skew partial identity detector improves the inaccuracy of the computed clock skew for the computer with IPv4 address *10.0.0.1* and sends the following *continue* message m_4:

- timestamp: 1494320500,
- identifiers: IPv4 address *10.0.0.1*, clock skew *20 ppm*,
- inaccuracy: 1 ($i := 1$).

According to Algorithm 8.2, as the corresponding *begin* message m_3 added the edge e_2 and $\Pi_4(\mathit{attributes}(e_2)) \neq i$, G is redefined as follows:

- Algorithm 8.2 does not modify V,
- $E := \{e_1, e_2, e_3\}$,
- $\mathit{endpts} := \{(e_1, \{\mathit{IPv4{:}\ 10.0.0.1}, \mathit{MAC{:}\ 00{:}11{:}22{:}33{:}44{:}55}\}),$
  $(e_2, \{\mathit{IPv4{:}\ 10.0.0.1}, \mathit{skew\ 20\,ppm}\}),$
  $(e_3, \{\mathit{IPv4{:}\ 10.0.0.1}, \mathit{skew\ 20\,ppm}\})\},$
- Algorithm 8.2 does not modify *category*,
- $\mathit{attributes} := \{(e_1, (\mathrm{DHCP}, 1494320000, \infty, 0)),$
  $(e_2, (\mathrm{clockskew}, 1494320302, 1494320500, 2)),$
  $(e_3, (\mathrm{clockskew}, 1494320500, \infty, 1))\}.$
- Additionally, remember that e_3 is the current edge for session started by the *begin* message m_3.

DHCP partial identity detector detects that the IPv4 address was released — the DHCP partial identity detector sends the following *end* message m_5:

- timestamp: 1494321000,
- identifiers: IPv4 address *10.0.0.1*, MAC address *00:11:22:33:44:55*,

The corresponding *begin* and *continue* messages for the session are m_1 and m_2. The only added edge for m_1 and m_2 is e_1. According to algorithm 8.3, *attributes* is redefined as follows:

- $\mathit{attributes} := \{(e_1, (\mathrm{DHCP}, 1494320000, 1494321000, 0)),$
  $(e_2, (\mathrm{clockskew}, 1494320302, 1494320500, 2)),$
  $(e_3, (\mathrm{clockskew}, 1494320500, \infty, 1))\}$

Clock skew partial identity detector detects a long inactivity of the computer with IPv4 address *10.0.0.1* and sends the following *end* message m_6:

- timestamp: 1494324600,
- identifiers: IPv4 address *10.0.0.1*, clock skew *20 ppm*,

The corresponding *begin* and *continue* messages for the session are m_3 and m_4 that added edges e_2 and e_3. However, attributes of e_2 were already changed as a consequence of the processing of the *continue* message m_4. According to Algorithm 8.3, *attributes* is redefined as follows:

- $\mathit{attributes} := \{(e_1, (\mathrm{DHCP}, 1494320000, 1494321000, 0)),$
  $(e_2, (\mathrm{clockskew}, 1494320302, 1494320500, 2)),$
  $(e_3, (\mathrm{clockskew}, 1494320500, 1494324600, 1))\}$

Note that information is only inserted to the identity graph and never deleted. Therefore, network forensic investigators can employ identity graphs during the investigation of past events.

B.2 PPP, RADIUS, and ND without application identifiers

This section provides examples of queries in an identity graph constructed based on PPP, RADIUS, and ND partial identity sources.

Suppose that information is gathered from PPP, RADIUS, and ND and the following information is revealed:

- *RADIUS log* contains that *JohnDoe* authenticated MAC address *aa:bb:cc:00:11:22* and received IPv4 address *10.0.0.1*. The authentication is valid from *2017-01-01T12:00* to *2017-01-01T13:00*.
- *ND tracking* reveals at *2017-01-01T12:01* that the network interface with MAC address *aa:bb:cc:00:11:22* generated IPv6 address *2001:db8::1*. The assignment is valid until *2017-01-01T12:31*.
- *PPP log* reveals that *JohnDoe* connected from MAC address *aa:bb:cc:55:66:77* through PPP and received IPv4 address *10.11.12.1*. The authentication is valid from *2017-01-01T12:45* to *2017-01-01T13:30*.
- *RADIUS log* reveals that *JohnDoe* authenticated IPv4 address *10.11.12.1*. The authentication is valid from *2017-01-01T12:45* to *2017-01-01T13:30*.

Figure B.1 shows the identity graph constructed from the revealed information.

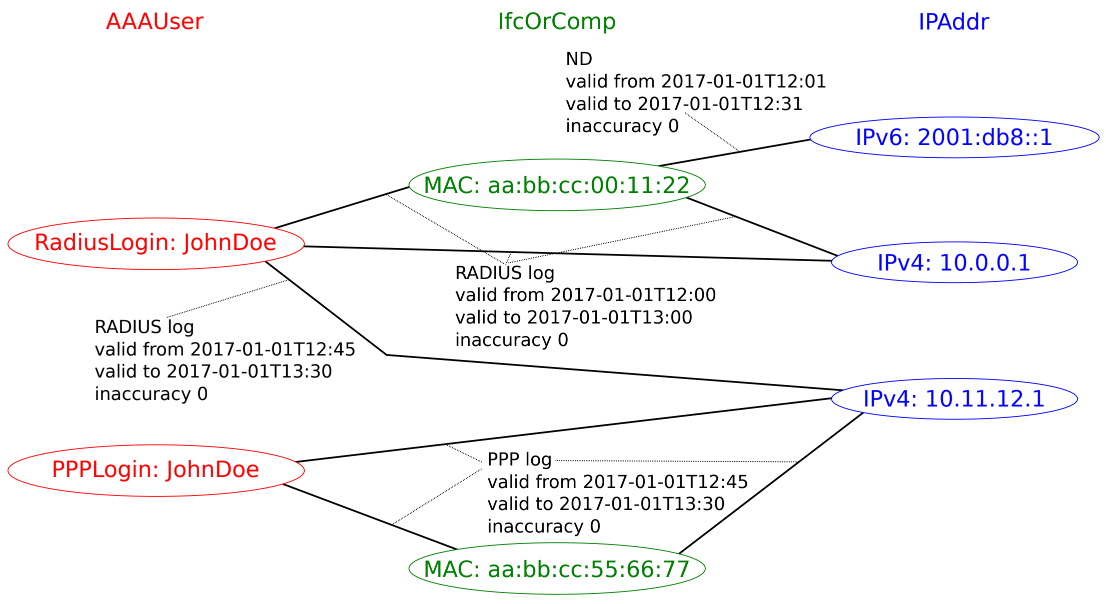


Figure B.1: The identity graph constructed from the RADIUS, PPP, and DHCPv6 logs.

- A query for all components of the partial identity represented by the RADIUS identifier *JohnDoe* can be computed as

$$\mathit{linked}(\text{"RadiusLogin: JohnDoe"}, \{l_{8.3}\}),$$

which yields MAC address *aa:bb:cc:00:11:22*, IPv4 address *10.0.0.1*, IPv4 address *10.11.12.1*, and IPv6 address *2001:db8::1*.

- A query for all components of the partial identity represented by the RADIUS identifier *JohnDoe* at time *2017-01-01T12:55* can be computed as

  $$\mathit{linked}(\text{"RadiusLogin: JohnDoe"}, \{l_{8.3}, l_{8.13}(\text{2017-01-01T12:55}, \text{2017-01-01T12:55})\}),$$

  which yields MAC address *aa:bb:cc:00:11:22*, IPv4 address *10.0.0.1*, and IPv4 address *10.11.12.1*.

- A query for all components of the partial identity represented by the RADIUS identifier *JohnDoe* active continuously between *2017-01-01T12:10* and *2017-01-01T12:55* can be computed as

  $$\mathit{linked}(\text{"RadiusLogin: JohnDoe"}, \{l_{8.3}, l_{8.13}(\text{2017-01-01T12:10}, \text{2017-01-01T12:55})\}),$$

  which yields MAC address *aa:bb:cc:00:11:22* and IPv4 address *10.0.0.1*.

- A query for all components of the partial identity represented by the RADIUS identifier *JohnDoe* active anytime between *2017-01-01T12:10* and *2017-01-01T12:55* can be computed as

  $$\mathit{linked}(\text{"RadiusLogin: JohnDoe"}, \{l_{8.3}, l_{8.14}(\text{2017-01-01T12:10}, \text{2017-01-01T12:55})\}),$$

  which yields MAC address *aa:bb:cc:00:11:22*, IPv4 address *10.0.0.1*, IPv4 address *10.11.12.1*, and IPv6 address *2001:db8::1*.

- A query for all components of the partial identity represented by the RADIUS identifier *JohnDoe* always linkable at *2017-01-01T12:10*, *2017-01-01T12:30*, and *2017-01-01T12:50* can be computed as

  $$\begin{aligned}\mathit{linked}_{\text{repeatedly}}(\text{"RadiusLogin: JohnDoe"}, \{&\text{2017-01-01T12:10}, \text{2017-01-01T12:30},\\ &\text{2017-01-01T12:50}\}\{l_{8.3}\})\end{aligned}$$

  which yields MAC address *aa:bb:cc:00:11:22* and IPv4 address *10.0.0.1*.

- A query for all components of the partial identity represented by the RADIUS identifier *JohnDoe* linkable at *2017-01-01T12:10*, *2017-01-01T12:30*, or *2017-01-01T12:50* can be computed as

  $$\begin{aligned}\mathit{linked}_{\text{anytime}}(\text{"RadiusLogin: JohnDoe"}, \{&\text{2017-01-01T12:10}, \text{2017-01-01T12:30},\\ &\text{2017-01-01T12:50}\}\{l_{8.3}\}),\end{aligned}$$

  which yields MAC address *aa:bb:cc:00:11:22*, IPv4 address *10.0.0.1*, IPv4 address *10.11.12.1*, and IPv6 address *2001:db8::1*.

- A query for all components of the partial identity represented by the IPv4 address *10.0.0.1* can be computed as

  $$\mathit{linked}(\text{"IPv4: 10.0.0.1"}, \{l_{8.3}\}),$$

  which yields an empty set.

- A query for all identifiers of partial identities of the computer identified by IPv4 address *10.0.0.1* can be computed as

$$linked(\text{"IPv4: 10.0.0.1"}, \{l_{8.5}\}),$$

which yields MAC address *aa:bb:cc:00:11:22* and IPv6 address *2001:db8::1*.

- A query for identifiers of all partial identities of computers where *JohnDoe* authenticated through RADIUS can be computed as

$$linked(\text{"RadiusLogin: JohnDoe"}, \{l_{8.7}\}),$$

which yields MAC address *aa:bb:cc:00:11:22*, MAC address *aa:bb:cc:55:66:77*, IPv4 address *10.0.0.1*, IPv4 address *10.11.12.1*, and IPv6 address *2001:db8::1*.

- A query for all user accounts logged in or authenticated from a computer identified by IPv4 address *10.11.12.1* can be computed as

$$linked(\text{"IPv4: 10.11.12.1"}, \{l_{8.10}\}),$$

which yields PPP login *JohnDoe* and RADIUS login *JohnDoe*.

Note that the input vertex is never returned by *linked*. This is because this thesis considers neither paths that do not contain any edge nor loops. Nevertheless, it is trivial to define functions that insert the input identifier to the resulting set, should the input identifier be required to be returned by an operation.

B.3 Linkage of IRC-related information

This section shows two examples of identity linking related to application layer identifiers. Both examples show IRC graphs created by information from a partial identity detector that parses IRC protocol.

The first example shows two users entering the same IRC network and joining the same IRC channel *foo*. For simplicity, this example intentionally ignores time-related information. Suppose that the IRC traffic parser reveals the following information:

- The computer using IPv4 address *192.168.1.1* connects to the IRC server at *172.16.1.7*, local port 1122, server port 7000. The user selects nickname *Alice*.
- The computer using IPv4 address *10.0.0.1* connects to the IRC server at *172.16.1.7*, local port 1234, server port 7000. The user selects nickname *Bob*.
- Both *Alice* and *Bob* join channel *foo* and participate in a discussion in the channel.

Figure B.2 shows the identity graph created using the learnt information. Note that each nickname is connected to the client IP address to reflect that the user connected from that IP address. The nicknames are not connected to the server IP address.

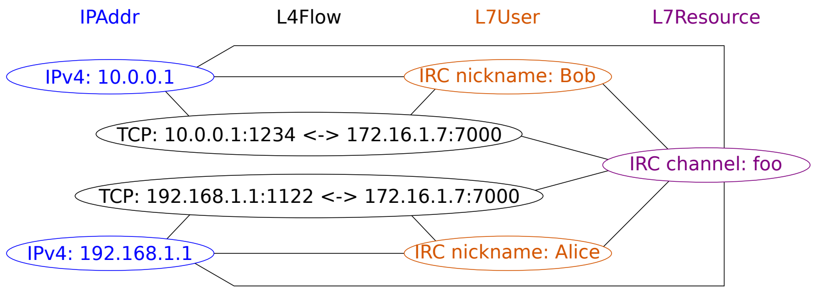


Figure B.2: The identity graph constructed by monitoring of IRC traffic (first example).

- A query for all identifiers of all computers where *Bob* was logged to an IRC network can be computed as

$$\mathit{linked}(\text{"IRC nickname: Bob"}, \{l_{8.7}\}),$$

which yields IPv4 address *10.0.0.1* and TCP flow *10.0.0.1:1234* $\leftrightarrow$ *172.16.1.7:7000*.

- A query for all usernames of all users that joined channel *foo* can be computed as

$$\mathit{linked}(\text{"IRC channel: foo"}, \{l_{8.8}\}),$$

which yields both nicknames, *IRC nickname: Alice* and *IRC nickname: Bob*.

- A query for all usernames connecting from the computer identified by IPv4 address *192.168.1.1* can be computed as

$$\mathit{linked}(\text{"IPv4: 192.168.1.1}, \{l_{8.10}\}),$$

which yields *IRC nickname: Alice*.

- A query for resources accessed from the computer identified by IPv4 address *192.168.1.1* can be computed as

$$\mathit{linked}(\text{"IPv4: 192.168.1.1}, \{l_{8.12}\}),$$

which yields *IRC channel: foo*.

The second IRC example shows a single user accessing IRC network from two computers. Suppose that the IRC traffic parser reveals the following information:

- At time t_0, the computer using IPv4 address *192.168.1.1* connects to the IRC server at *172.16.1.7*, local port 1122, server port 7000. The user selects nickname *Alice*.
- At time t_1, *Alice* joins channel *foo* and participate in a discussion in the channel.
- At time t_2, *Alice* disconnects from the IRC network.
- At time t_3, the computer using IPv4 address *10.0.0.1* connects to the IRC server at *172.16.1.7*, local port 1234, server port 7000. The user selects nickname *Alice*.
- At time t_4, *Alice* joins channel *moo* and participate in a discussion in the channel.
- At time t_5, *Alice* disconnects from the IRC network.

Figure B.3 shows the identity graph created using the learnt information. Edge attributes except time validity are not displayed. Note that both the nickname and the channel names are connected to the client IP address to reflect that the user connected from that IP address and that the resource was accessed from that IP address.

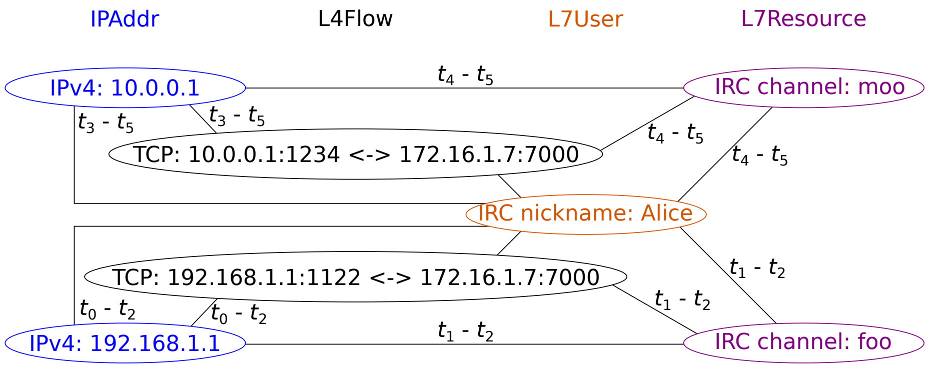


Figure B.3: The identity graph constructed by monitoring of IRC traffic (second example).

- A query for all components of the computer identified by IP address *10.0.0.1* can be computed as

  $$linked(\text{"IPv4: 10.0.0.1"}, \{l_{8.3}\}),$$

  which yields TCP flow *10.0.0.1:1234* $\leftrightarrow$ *172.16.1.7:7000.*

- A query for all partial identities of the same computer represented by IP address *10.0.0.1* can be computed as

  $$linked(\text{"IPv4: 10.0.0.1"}, \{l_{8.5}\}),$$

  which yields TCP flow *10.0.0.1:1234* $\leftrightarrow$ *172.16.1.7:7000.*

- A query for all identifiers of all computers where *Alice* was logged to an IRC network can be computed as

  $$linked(\text{"IRC nickname: Alice"}, \{l_{8.7}\}),$$

  which yields both IPv4 addresses and both TCP flows.

- A query for all identifiers of users accessing *IRC channel: foo* can be computed as

  $$linked(\text{"IRC channel: foo"}, \{l_{8.8}\}),$$

  which yields *IRC nickname: Alice.*

- A query for all usernames that connected from the computer identified by IPv4 address *192.168.1.1* can be computed as

  $$linked(\text{"IPv4: 10.0.0.1}, \{l_{8.10}\}),$$

  which yields *IRC nickname: Alice.*

- A query for resources accessed from the computer identified by IPv4 address *10.0.0.1* can be computed as
$$\mathit{linked}(\text{"IPv4: 10.0.0.1}, \{l_{8.12}\}),$$
which yields *IRC channel: moo*.

- A query for resources accessed from the computer identified by IPv4 address *10.0.0.1* at time t_5 can be computed as
$$\mathit{linked}(\text{"IPv4: 10.0.0.1}, \{l_{8.12}, l_{8.13}(t_5, t_5)\}),$$
which yields *IRC channel: moo*.

- A query for resources accessed by *Alice* at time t_1, t_3, or t_5 can be computed as
$$\mathit{linked}_{\text{anytime}}(\text{"IRC nickname: Alice}, \{(t_1, t_1), (t_3, t_3), (t_5, t_5)\}\{l_{8.12}\}),$$
which yields both *IRC channel: foo* and *IRC channel: moo*.

B.4 Inaccuracy

This example shows an identity graph created from clock skew estimation. For simplicity, this example shows only inaccuracy as an edge attribute. The example does not show any time-related query and the partial identity detector of all edges is *clock skew*. Suppose that the clock skew partial identity detector provides the following information:

- The detected clock skew for IPv6 address *2001:db8::1* is 23.7 ± 0.5 ppm.
- The detected clock skew for IPv6 address *2001:db8::2* is 23.9 ± 0.8 ppm.
- The detected clock skew for IPv6 address *2001:db8::3* is 25.3 ± 1.0 ppm.
- The detected clock skew for IPv6 address *2001:db8::4* is -10.7 ± 0.5 ppm.

The mechanism detected 4 IPv6 addresses (*IPAddr* identifiers) and 4 clock skew (hidden identifiers of category *IfcOrComp*). The uncertainty of clock skew measurement means that it is possible that more clock skew identifiers identify an IPv6 address.

When determining the inaccuracy of linking a clock skew estimation with an IP address, for *IPAddr* identifier, let us denote the minimal estimated clock skew of the computer using an IP address as i_{min}, the maximal estimated clock skew of the computer using an IP address as i_{max}. For clock skew value *IfcOrComp* identifier, let us denote the minimal clock skew estimation value as c_{min}, and the maximal clock skew estimation value as c_{max}.

For this example, let us use the function I defined in Formula B.1 to determine the inaccuracy for an edge between an overlapping clock skew identifier and a clock skew estimation of the computer using an IP address. Note that the function was chosen by educated guess and there is no research evaluating the suitability of the function.

$$I \equiv \frac{\max(c_{max}, i_{max}) - min(c_{min}, i_{min})}{\min(c_{max}, i_{max}) - max(c_{min}, i_{min})} \tag{B.1}$$

Note that the minimal value of I is 1, reflecting that the clock skew estimation is not completely accurate partial identity detector (see Chapter 7 for more details on the accuracy).

Figure B.4 displays the learnt information and the computed inaccuracies.

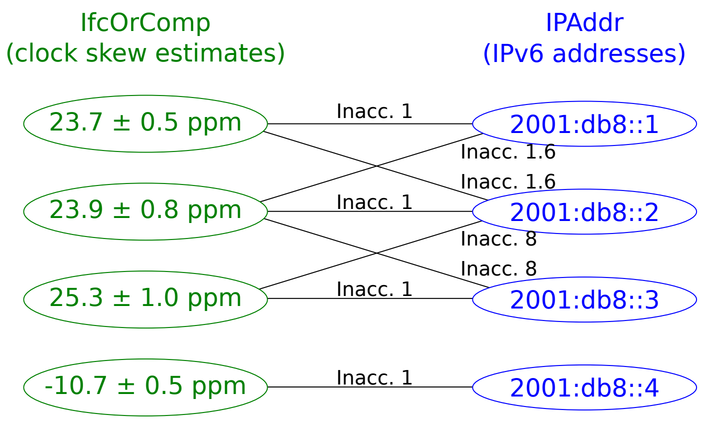


Figure B.4: The identity graph constructed for the clock skew monitoring example.

- A query for all linkable identifiers of *2001:db8::1* with the maximal inaccuracy of 5 can be computed as

$$\mathit{linked}(\text{"2001:db8::1"}, \{l_{8.19}(5)\}),$$

  which yields IPv6 address *2001:db8::2* and clock skew estimates $23.7 \pm 0.5\,\mathit{ppm}$ and $23.9 \pm 0.8\,\mathit{ppm}$.

- Changing the inaccuracy to 10:

$$\mathit{linked}(\text{"2001:db8::1"}, \{l_{8.19}(10)\})$$

  yields in addition to the previous case the IPv6 address *2001:db8::3*.

- By increasing the inaccuracy further to 11:

$$\mathit{linked}(\text{"2001:db8::1"}, \{l_{8.19}(11)\})$$

  yields also the clock skew identifier $25.3 \pm 1.0\,\mathit{ppm}$.

The inaccuracy defined by Formula B.1 and queried inaccuracy have to be tweaked for the deployment on a particular site.

B.5 NAT

This example shows operations in a network with NAT. Let partial identity detectors provide the following knowledge:

- *SIP traffic analysis* reveals that *Alice* connected to the SIP server with the flow *TCP: 1.2.3.4:1234* $\leftrightarrow$ *147.229.1.1:5060* (f_1) and she makes a call in the flow *UDP: 1.2.3.4: 5678* $\leftrightarrow$ *5.6.7.8:2233* (f_2).

- *NAT tracking* reveals that the flow f_1 is translated from the flow *TCP: 10.0.0.1:1234* $\leftrightarrow$ *147.229.1.1:5060* (f_3).

- *NAT tracking* reveals that the flow f_2 is translated from flow f_4.

- *NAT tracking* additionally reveals another translation of a flow originating from another computer. The flow *TCP: 10.0.0.2:11223* $\leftrightarrow$ *11.12.13.14:80* (f_5) is translated to *TCP: 1.2.3.4:11223* $\leftrightarrow$ *11.12.13.14:80* (f_6).

Figure B.5 shows the resulting identity graph without the modifications defined in Subsection 8.4.5. Notice that a query for all partial identities of computers where *Alice* authenticated or logged in

$$\textit{linked}(\text{"SIP account: Alice"}, \{l_{8.7}\})$$

yields the set containing IPv4 address *1.2.3.4* and all flows f_1, f_2, f_3, f_4, f_5, and f_6 even though f_5 and f_6 does not belong to the computer where Alice was logged in.

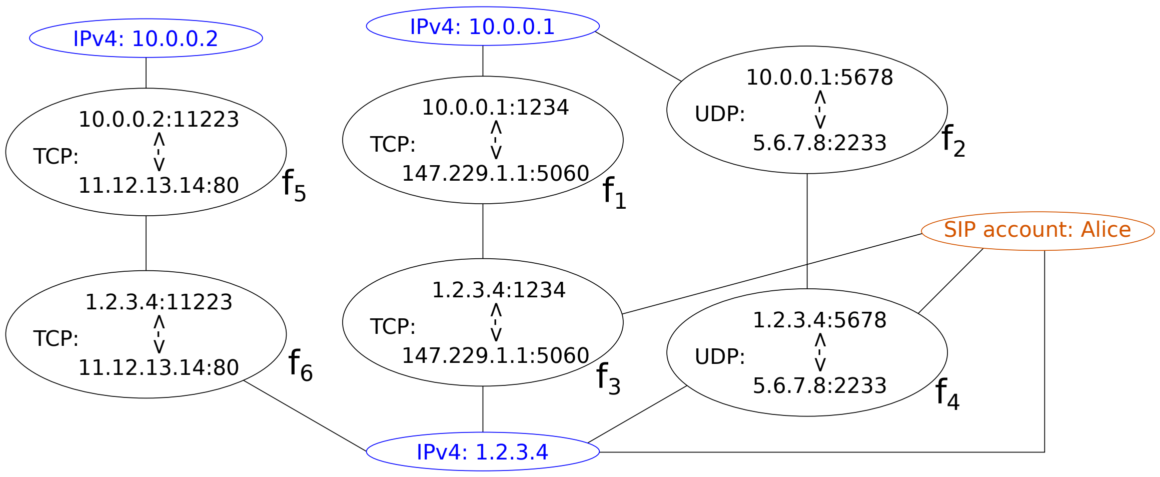


Figure B.5: The identity graph constructed for the network with NAT without the rules defined by Section 8.4.5.

Figure B.6 shows the identity graph constructed according to the rules presented in Subsection 8.4.5. In this case,

$$\textit{linked}(\text{"SIP account: Alice"}, \{l_{8.7}\})$$

yields the set containing IPv4 address *10.0.0.1* and flows f_1, f_2, f_3, and f_4. The IP address and all flows belong to the computer from which Alice connected to the SIP account.

Note that it is possible to query all partial identities of the network address translator by

$$\textit{linked}(\text{"IPv4: 1.2.3.4"}, \{l_{8.3}\}),$$

which yields flows f_1, f_2, f_3, f_4, f_5, and f_6.

B.6 CGN

This example shows operations in a network with CGN. Let partial identity detectors provide the following knowledge:

- *SIP traffic analysis* reveals that *Alice* connected to the SIP server in the flow *TCP: 1.2.3.4:1234* $\leftrightarrow$ *147.229.1.1:5060* (f1).

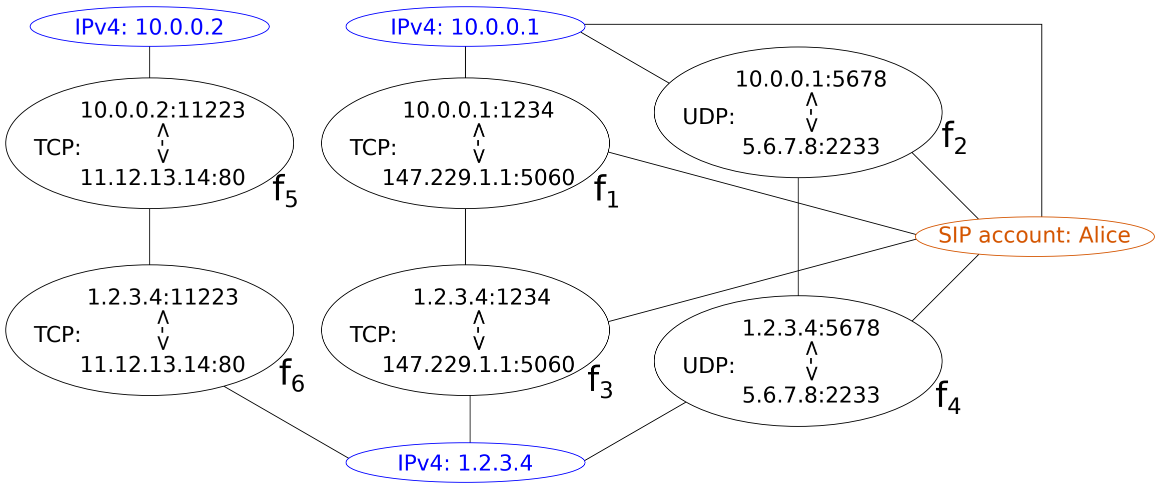


Figure B.6: The identity graph constructed for the network with NAT as defined by Subsection 8.4.5.

- *CGN tracking* reveals that the flow f1 is translated from the flow *TCP: 100.64.0.1:1234* $\leftrightarrow$ *147.229.1.1:5060* (f2).
- *NAT tracking* reveals that the flow f2 is translated from the flow *TCP: 10.0.0.2:1234* $\leftrightarrow$ *147.229.1.1:5060* (f3).
- *CGN tracking* reveals that the flow *TCP: 1.2.3.5:5678* $\leftrightarrow$ *195.113.1.1:80* (f4) is translated from the flow *TCP: 100.64.0.1:5678* $\leftrightarrow$ *195.113.1.1:80* (f5).
- *NAT tracking* reveals that the flow f5 is translated from the flow *TCP: 10.0.0.2:5678* $\leftrightarrow$ *195.113.1.1:80* (f6).
- *SIP traffic analysis* reveals that *Bob* connected to the SIP server with the flow *TCP: 1.2.3.4:2222* $\leftrightarrow$ *147.229.1.1:5060* (f7).
- *CGN tracking* reveals that the flow f7 is translated from the flow *TCP: 100.64.0.2:2222* $\leftrightarrow$ *147.229.1.1:5060* (f8).

Figure B.7 shows the identity graph created based on the knowledge; the rules specified in Subsection 8.4.5 are applied.

- A query for all components of the computer identified by IP address *10.0.0.2* can be computed as

  $$linked(\text{"IPv4: 10.0.0.2"}, \{l_{8.3}\}),$$

  which yields TCP flows f1, f2, f3, f4, f5, and f6.
- A query for all partial identities of the computer represented by IP address *10.0.0.2* can be computed as

  $$linked(\text{"IPv4: 10.0.0.2"}, \{l_{8.5}\}),$$

  which yields TCP flows f1, f2, f3, f4, f5, and f6.

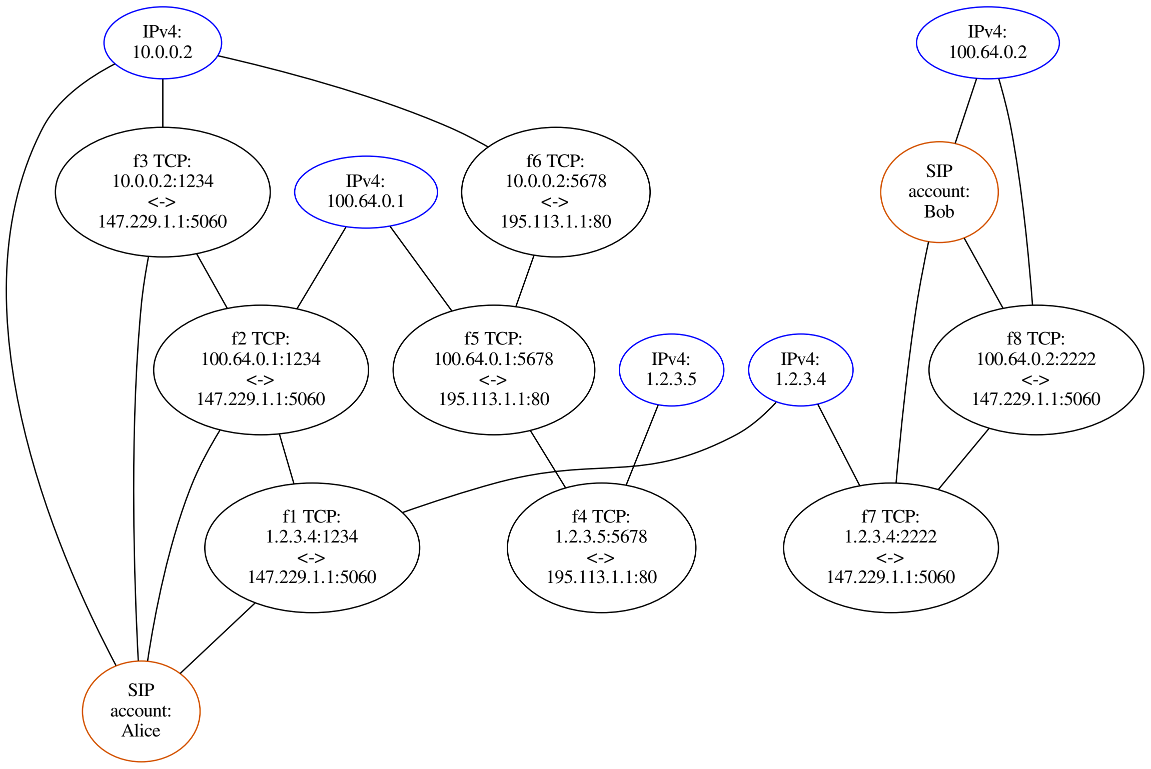

IPv4: 10.0.0.2
IPv4: 100.64.0.2
f3 TCP: 10.0.0.2:1234 <-> 147.229.1.1:5060
IPv4: 100.64.0.1
f6 TCP: 10.0.0.2:5678 <-> 195.113.1.1:80
SIP account: Bob
f2 TCP: 100.64.0.1:1234 <-> 147.229.1.1:5060
f5 TCP: 100.64.0.1:5678 <-> 195.113.1.1:80
IPv4: 1.2.3.5
IPv4: 1.2.3.4
f8 TCP: 100.64.0.2:2222 <-> 147.229.1.1:5060
f1 TCP: 1.2.3.4:1234 <-> 147.229.1.1:5060
f4 TCP: 1.2.3.5:5678 <-> 195.113.1.1:80
f7 TCP: 1.2.3.4:2222 <-> 147.229.1.1:5060
SIP account: Alice

Figure B.7: The identity graph constructed for the CGN example.

- A query for all usernames that connected from the computer identified by IPv4 address *10.0.0.2* can be computed as

$$linked(\text{"IPv4: 10.0.0.2}, \{l_{8.10}\}),$$

which yields *IRC nickname: Alice*.

- A query for all components of the partial identity represented by SIP username *Alice* can be computed as

$$linked(\text{"SIP account: Alice"}, \{l_{8.3}\}),$$

which yields TCP flows f1, f2, and f3.

- A query for all identifiers of all computers where *Alice* was logged to her SIP account can be computed as

$$linked(\text{"SIP account: Alice"}, \{l_{8.7}\}),$$

which yields IPv4 address *10.0.0.2* and TCP flows f1, f2, f3, f4, f5, and f6.

- A query for all identifiers of all computers where *Bob* was logged to his SIP account can be computed as

$$linked(\text{"SIP account: Bob"}, \{l_{8.7}\}),$$

which yields IPv4 address *100.64.0.2* and TCP flows f7 and f8.

Appendix C

Deployment of identity graphs

This appendix depicts networks, in which identity graphs were deployed and tested. The goal of each testing network was to validate that deployed partial identity detectors reveals all partial identities, and it is possible to construct identity graphs and link detected identifiers to IPv4 and IPv6 addresses. When identity graphs were deployed as a part of SLIS, SLIS configured Sec6Net probes to intercept traffic of suspects in the network.

Small controlled networks in a laboratory

Figure C.1 shows a controlled network used for testing DHCP and RADIUS partial identity detectors that observe traffic at the IAP. Computers in the LAN obtain IPv4 configuration via DHCP, the router relays DHCP traffic to the DHCP server. Additionally, computers in the LAN authenticate via RADIUS.

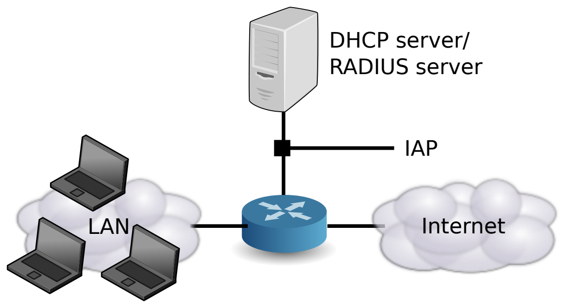


Figure C.1: An interception of an IPv4 LAN.

Networks similar to the network depicted in Figure C.1 were tested to detect partial identities in DHCP, DHCPv6, RADIUS and SLAAC. SLIS was typically deployed at the IAP.

DSL networks

Figure C.2 depicts a testing network with *Digital subscriber line* (DSL). Modems connect subscribers to the ISP network through *DSL access multiplexer* (DSLAM) and *broadband remote access server* (BRAS). There are telephone links between modems and DSLAM. PPPoE [120] is deployed between modems and BRAS. Modems authenticate to the BRAS using PPPoE. Depending on the test cases, BRAS can be configured to relay the authentication to the RADIUS server. Some modems support IPv4 only and obtain a single IPv4 address from PPPoE. Some modems support IPv6 and lease prefixes using DHCPv6.

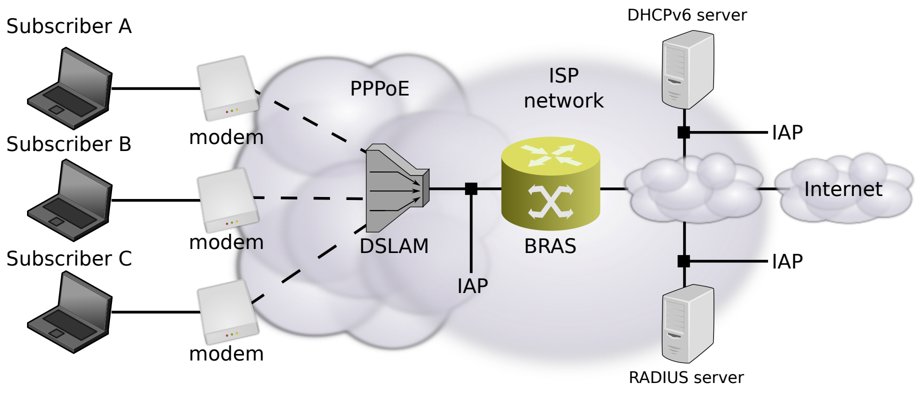


Figure C.2: Network containing DSL subscribers.

During testing in the network depicted in Figure C.2, we focused on the detection and linking of partial identities obtained from:

- PPPoE (IAP between DSLAM and BRAS) — MAC addresses, usernames, IPv4, and IPv6 addresses.
- DHCPv6 (IAP close to DHCPv6 server) — DHCPv6 prefix delegation [186].
- RADIUS (optional IAP close to the RADIUS server) — usernames, MAC addresses, IPv4 addresses.

SLIS was connected to the ISP network. Partial identity detectors were distributed at the IAPs.

Permanent testing network

Figure C.3 a permanent testing network that was used for testing identity detectors developed during this PhD research and the Sec6Net project. There were static IPv6 addresses, DHCP and DHCPv6 server and relay permanently operating in the network. Additionally, the router advertised IPv6 prefix. During testing, clients and partially identity detectors including application layer detectors were deployed on the PCs.

Deployment in faculty network

Figure C.4 shows the monitoring deployment that was also used to validate the IPv6 address assignment tracking discussed in Chpter 6. The monitoring PC was connected to the faculty network and was able to observe ND traffic. Consequently, the IPv6 address tracking partial identity detector observed addresses assigned to other computers in the faculty network. Additionally, the monitoring node observed copy of the DHCP traffic of three computers in our office. Subsection 6.5.4 discusses the IPv6 address assignments monitoring.

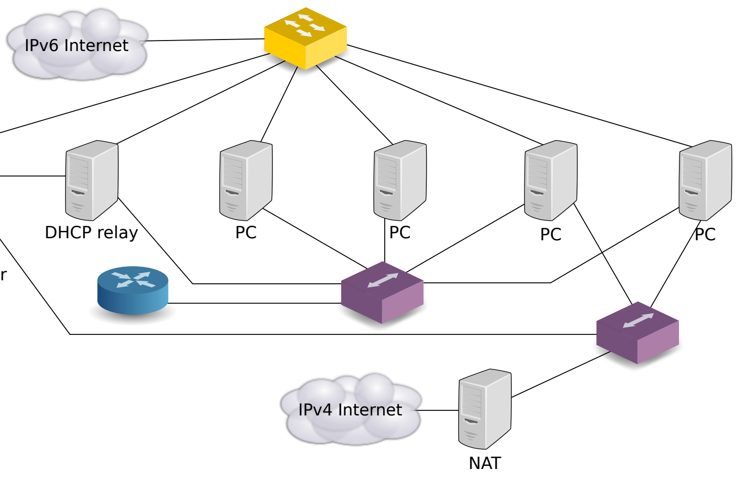


Figure C.3: Permanent testing network used to test partial identity detectors developed during this PhD research.

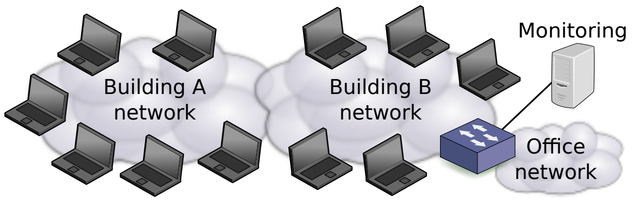


Figure C.4: Real network monitoring.

Stress testing

Figure C.5 shows a network used for stress testing of the IPv6 address assignments tracking described in Chapter 6. The packet generator transmitted traffic simulating a predefined number of devices in a local network. Each device generated a particular IPv6 address and performed DAD for the address. As the MAC addresses and generated IPv6 addresses were predictable, we were able to determine that all devices simulated by Spirent appeared in the identity graph.

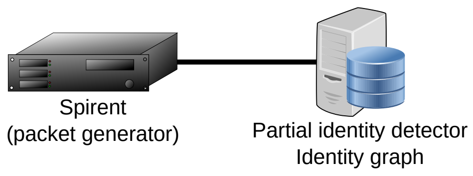


Figure C.5: Stress testing deployment.

The same network was also used to test SMTP partial identity detector [91]. Spirent simulated both SMTP clients and servers. Identity graph that included e-mail addresses and IP addresses was constructed based on the messages from the SMTP partial identity detector.

Demonstration networks for Czech LEA

Figure C.6 shows one of the topologies with the 1 Gbps Sec6Net CC-IIF probe. DHCP server leases IPv4 addresses to devices in the network, the router advertises IPv6 prefix. The WiFi access point performs as a bridge between the Ethernet network and laptops. The DHCP and ND (see Chapter 6) partial identity detectors detect the IP addresses used by the laptops. During the demonstration, LEA officers were able to connect to the network, obtain IP addresses and access the web server. The partial identity detectors detected IPv4, IPv6, and MAC addresses and the identity graph linked the identifiers. Based on the detected partial identities, it was possible to configure the Sec6Net probe and intercept traffic of selected laptops.

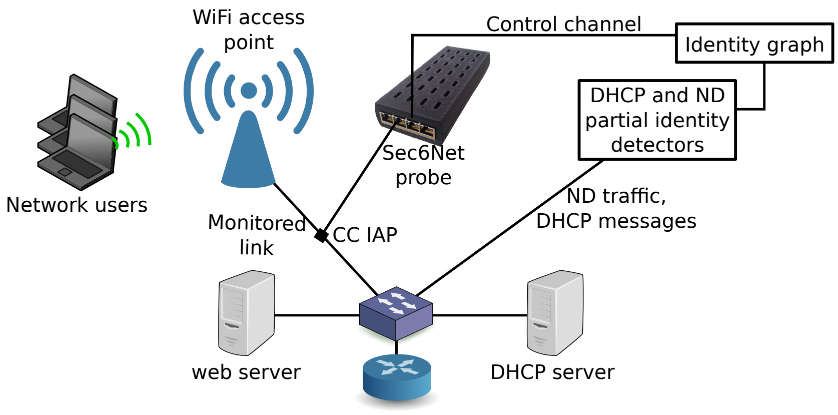


Figure C.6: An example of a demonstration network for Czech LEA.

Figure C.7 depicts a network that includes application layer partial identity detectors, namely SMTP and SIP. The goal of the demonstration was to generate 10 Gbps of traffic that included SMTP and SIP traffic, SMTP and SIP partial identity detectors were deployed on the Sec6Net probe. The identity graph was build based on identifiers detected by partial identity detectors. It was possible to capture traffic based on the application identifiers — SIP usernames and e-mail addresses.

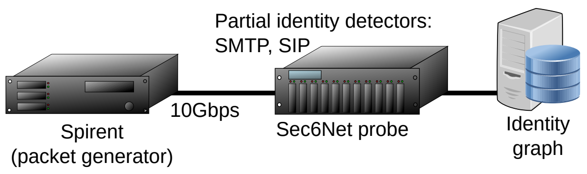

Partial identity detectors: SMTP, SIP
10Gbps
Spirent (packet generator)
Sec6Net probe
Identity graph

Figure C.7: Testing of partial identity detectors running on a high speed Sec6Net probe.

The goal of the demonstrations is to show that partial identity detectors can be distributed on the network and located at the most suitable place, while the computer building the identity graph can be deployed at the most convenient place. Partial identity detectors running on Sec6Net probes can be deployed in high-speed networks.