**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

Ertugrul Dogruluk

# Side-Channel Timing Attack on Content Privacy of Named Data Networking

november 2020

Ertugrul Dogruluk **Side-Channel Timing Attack on Content Privacy of Named Data Networking**

Uminho | 2020

**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

Ertugrul Dogruluk

# Side-Channel Timing Attack on Content Privacy of Named Data Networking

Doctoral Thesis
Doctor of Philosophy Degree in Electronics and Computer Engineering

Thesis supervised by
**Joaquim Macedo**
**Antonio Costa**

november 2020

## DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos. Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada. Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

**Atribuição-NãoComercial-SemDerivações**
**CC BY-NC-ND**
https://creativecommons.org/licenses/by-nc-nd/4.0/

## Acknowledgments

First of all, I would like to express my thanks to Professor Joaquim Macedo and Professor Antonio Costa, for supervising this study.

I would like to thank Professor Joaquim Macedo for his guidance, support, and valuable scientific advice in order to contribute to this study. His scientific suggestions used to improve this study, especially guiding the right research direction.

I would like to thank Professor Antonio Costa for his technical support and scientific advice for any step of this study. He supervised the scenario implementation related works and suggested the scientific contributions for this study.

I have enjoyed working with both of them. They spent a lot of time with me for revising this thesis in order to improve the quality of this study. Without their effort and support, this work would never be achieved, especially in a pandemic period.

I had useful discussions with Óscar Gama, who is a postdoctoral researcher member of Computer Communications, and Networks (CCN) lab. I'm so grateful to get his support and his given scientific ideas for my research.

I thank Professor Alexandre Santos and members of the Department of Informatics to support the laboratory environment and research facilities for this study.

To my family, especially my parents Yasar and Sebahat Dogruluk, for the unconditional support, love, and understanding during these years that I have been away from home.

*Thank you all for supporting me to archiving my goal.*

## Statement of Integrity

I hereby declare having conducted this academic work with integrity.
I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.
I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

# ABSTRACT

A diversity of current applications, such as Netflix, YouTube, and social media, have used the Internet mainly as a content distribution network. Named Data Networking (NDN) is a network paradigm that attempts to answer today's applications need by naming the content. NDN promises an optimized content distribution through a named content-centric design. One of the NDN key features is the use of in-network caching to improve network efficiency in terms of content distribution. However, the cached contents may put the consumer privacy at risk. Since the time response of cached contents is different from un-cached contents, the adversary may distinguish the cached contents (targets) from un-cached ones, through the side-channel timing responses. The scope of attack can be towards the content, the name, or the signature. For instance, the adversary may obtain the call history, the callee or caller location on a trusted Voice over NDN (VoNDN) and the popularity of contents in streaming applications (*e.g.* NDNtube, NDNlive) through side-channel timing responses of the cache.

The side-channel timing attack can be mitigated by manipulating the time of the router responses. The countermeasures proposed by other researches, such as additional delay, random/probabilistic caching, group signatures, and no-caching can effectively be used to mitigate the attack. However, the content distribution may be affected by pre-configured countermeasures which may go against the goal of the original NDN paradigm. In this work, the detection and defense (DaD) approach is proposed to mitigate the attack efficiently and effectively. With the DaD usage, an attack can be detected by a multi-level detection mechanism, in order to apply the countermeasures against the adversarial faces. Also, the detections can be used to determine the severity of the attack. In order to detect the behavior of an adversary, a brute-force timing attack was implemented and simulated with the following applications and testbeds: *i.* a trusted application that mimics the VoNDN and identifies the cached certificate on a worldwide NDN testbed, and *ii.* a streaming-like NDNtube application to identify the popularity of videos on the NDN testbed and AT&T company. In simulation primary results showed that the multi-level detection based on DaD mitigated the attack about 39.1% in best-route, and 36.6% in multicast communications. Additionally, the results showed that DaD preserves privacy without compromising the efficiency benefits of in-network caching in NDNtube and VoNDN applications.

# RESUMO

Várias aplicações atuais, como o Netflix e o YouTube, têm vindo a usar a Internet como uma rede de distribuição de conteúdos. O Named Data Networking (NDN) é um paradigma recente nas redes de comunicações que tenta responder às necessidades das aplicações modernas, através da nomeação dos conteúdos. O NDN promete uma otimização da distribuição dos conteúdos usando uma rede centrada nos conteúdos. Uma das características principais do NDN é o uso da cache disponivel nos nós da rede para melhorar a eficiência desta em termos de distribuição de conteúdos. No entanto, a colocação dos conteúdos em cache pode colocar em risco a privacidade dos consumidores. Uma vez que a resposta temporal de um conteúdo em cache é diferente do de um conteúdo que não está em cache, o adversário pode distinguir os conteúdos que estão em cache dos que não estão em cache, através das respostas de side-channel. O objectivo do ataque pode ser direcionado para o conteúdo, o nome ou a assinatura da mensagem. Por exemplo, o adversário pode obter o histórico de chamadas, a localização do callee ou do caller num serviço seguro de voz sobre NDN (VoNDN) e a popularidade do conteúdos em aplicações de streaming (*e.g.* NDNtube, NDNlive) através das respostas temporais de side-channel.

O side-channel timing attack pode ser mitigado manipulando o tempo das respostas dos routers. As contramedidas propostas por outros pesquisadores, tais como o atraso adicional, o cache aleatório /probabilístico, as assinaturas de grupo e não fazer cache, podem ser efetivamente usadas para mitigar um ataque. No entanto, a distribuição de conteúdos pode ser afetada por contramedidas pré-configuradas que podem ir contra o propósito original do paradigma NDN. Neste trabalho, a abordagem de detecção e defesa (DaD) é proposta para mitigar o ataque de forma eficiente e eficaz. Com o uso do DaD, um ataque pode ser detectado por um mecanismo de detecção multi-nível, a fim de aplicar as contramedidas contra as interfaces dos adversários. Além disso, as detecções podem ser usadas para determinar a gravidade do ataque. A fim de detectar o comportamento de um adversário, um timing attack de força-bruta foi implementado e simulado com as seguintes aplicações e plataformas (testbeds): *i.* uma aplicação segura que implementa o VoNDN e identifica o certificado em cache numa plataforma NDN mundial; e *ii.* uma aplicação de streaming do tipo NDNtube para identificar a popularidade de vídeos na plataforma NDN da empresa AT&T. Os resultados da simulação mostraram que a detecção multi-nível oferecida pelo DaD atenuou o ataque cerca de 39,1% em best-route e 36,5% em comunicações multicast. Para avaliar o efeito nos pedidos legítimos, comparou-se o DaD com uma contramedida estática, tendo-se verificado que o DaD foi capaz de preservar todos os pedidos legítimos.

## CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS

| | |
|---|---|
| **ANDaNA** | Anonymous Named Data Networking Application |
| **CA** | Certificate Authority |
| **CCN** | Content Centric Network |
| **CDN** | Content Delivery Network |
| **CHR** | Cache Hit Ratio |
| **CRT** | Content Retrieval Time |
| **CRL** | Certificate Revocation List |
| **CS** | Content Store |
| **DNS** | Domain Name Server |
| **DON** | Data-Oriented Network |
| **DoS** | Denial of service |
| **FIB** | Forwarding Interest base |
| **FIFO** | First In First Out |
| **ICN** | Information-Centric Network |
| **IETF** | Internet Engineering Task Force |
| **IKB** | Interest-key Binding |
| **IoT** | Internet of Things |
| **ISP** | Internet Service Provider |
| **IP** | Internet Protocol |
| **LFU** | Least Frequently Used |
| **LRU** | Least Recently Used |
| **NACK** | Negative Acknowledgment |
| **NDN** | Named Data Network |
| **NFD** | Named Data Network Forwarding Daemon |
| **NLSR** | Named Data Network Link-State Routing Protocol |
| **OCSP** | Online Certificate Status Protocol |
| **OSPF** | Open Shortest Path First |
| **PIT** | Pending Interest Base |
| **PPKD** | Publisher Public Key Digest |
| **RDR** | Real-Time data Retrieval |
| **RIB** | Routing Information Base |
| **RTT** | Round-Trip Time |
| **RTSD** | Real-Time Streaming Data |
| **SCN** | Self-certifying Content Name |
| **SIP** | Session Initiation Protocol |
| **TCP** | Transmission Control Protocol |
| **TLS** | Transport Layer Security |
| **TLV** | Type-Length-Value |
| **UDP** | User datagram protocol |
| **URL** | Uniform Resource Locator |
| **VoIP** | Voice over Internet Protocol |
| **VoNDN** | Voice over Named Data Networks |

# 1

# INTRODUCTION

This chapter starts with a contextualization on Named Data Networks, as a new network architecture based on the emerging content-centric paradigm. Then the motivation and objectives of this work are presented, followed by the research methodology used. The major contributions of this work are also summarized.

## 1.1 Context

The Internet is being reshaped to handle content production and distribution, as nowadays users desire, for example, to watch movies and use social networking. Moreover, the number of IoT (Internet of Things) devices over the Internet is increasing enormously. However, such activities may not be the most appropriate to be done over the Internet, because this network was conceived for point-to-point communications. To overcome the problems raised by this communication paradigm, Content-Centric Networks (CCN) have been proposed. According to this new paradigm, replicas of content(s) are generated and cached. The aim of caching is to reduce the latency and data loss, thereby improving the distribution efficiency of popular content(s). For instance, the content-delivery networks (CDN) are (currently) used to replicate cache servers and increase content distribution. The Named Data Networks (NDN) paradigm was presented as the next version of CCN networks. The NDN is designed with a name-based network and cache (*e.g.* buffer-memory), presented by [16]. Through the in-network caching and named contents, the NDN promising a maximum content distribution compared with other CCN networks.

The NDN is based on a content-centric design, which does not require the content source and the destination address. Therefore, the NDN is supposed to provide better privacy because of the lack of source/destination addresses. However, the previously cached content, in spite of its benefits, may be targeted by side-channel timing attack to threaten the NDN privacy [16], [17], [18], [19]. Depending on the scope of the attack, the adversary may identify the location of producer and consumer by distinguishing cached and un-cached contents through the time differences from the cache.

Since the attack is based on time responses from the router cache, it can be mitigated by certain configurations based on the manipulation of the time. To mitigate the attack, static configurable countermeasure methods (delay, randomized cache, and encryption) were proposed by other researches [20], [17], [21], and [19]. However, any additional delay or name encryption may disable (reduced availability) of the cache, which can be considered as against in-networking caching based NDN design. Therefore, the statically configured countermeasures can be considered as a trade-off between the content distribution and privacy.

## 1.2 Motivation and Objectives

The in-network caching feature may arise the content privacy concerns in NDN. The adversary may identify the previously cached contents from which he can target sensitive information. In the side-channel timing attack, the adversary analyses the time responses from the cache of the router. The attacker can identify the producer, content name, and the certificate through the attack. The attack models were discussed by [17], [21], [18]. However, these can be considered as incapable to define the multiple targets and meet with the recent attack models.

The timing attack can be mitigated by an additional delay from the router cache response. Effectively, the statically configured countermeasure methods to mitigate the side-channel timing attack by serving the contents depending on various delay methods. Nevertheless, the countermeasure methods effectively mitigate the side-channel timing attack, they can reduce the performance of the NDN content distribution paradigm. For instance, any additional delay [17] into router content delivery may affect the content distribution efficiency because the legitimate requests are also affected by countermeasures. On the other hand, the contents must be cached in order to increase content distribution. Therefore, any additional delay to cache or completely disabled cache approaches ([21]) may be against the NDN paradigm.

Thus, this work was mainly motivated by overcoming the countermeasure distribution efficiency concerns, considered on the NDN paradigm. To achieve this main goal, the attack detection methods were presented to detect and distinguish the adversarial *face*[1] from other legitimate faces (consumers). Thus, a countermeasure method can be only applied to the adversary's face and the privacy and the distribution efficiency preserved dynamically.

Additionally, the detection methods can be used to obtain the severity of the attack. The attack can be classified as minor, moderate, and severe through the obtained detection metrics. Also, multiple countermeasure methods can be applied by the severity of attack definition respectively. For instance, the cache can be turned off on the adversary's face when a severe attack is detected on it, which can also be used to terminate the attack completely.

On the other hand, the scope of the attack and detection mechanisms may be different depending on NDN applications. For instance, the certificate privacy can be important on trust-based applications such as two-way communication application and name privacy can be important considering the video/audio streaming

[1] In NDN, the *interfaces* are called as *face*.

applications in NDN. Therefore, the attack and detection mechanisms can be modeled accordingly in NDN applications such as NDNtube and VoNDN.

As the motivation of works is identified above, this research is formed by three main research questions can be defined by the following:

A. *What kind of attack design can be achieved to increase the attack success?*

B. *Is it possible to mitigate the attack with the current countermeasures without compromising the NDN cache-store efficiency?*

C. *Is it possible to detect an adversary face to apply multiple countermeasure methods on different NDN applications?*

To answer these research questions above, this work addresses the following aspects, namely: *i.* The side-channel timing attack configuration and its threats to consumer/producer privacy, *ii.* Use the countermeasures only under the attack to the particular attack detected face. *iii.* Understand the adversary behavior on NDN applications and obtain the attack detection. Hence,

*The main objective of this work is the reduction in the impact of side-channel timing attack countermeasure methods on NDN content distribution performance.*

The side-channel timing attack is a threat to cached content in NDN. At the same time, it can be mitigated by countermeasure approaches as described by literature. However, these approaches may reduce the NDN content distribution efficiency while protecting the cache. To overcome this problem, the attack detection methods are evaluated to detect the attack. Through this strategy, efficiency can be maximized while mitigating the attack.

Further objectives are identified by the followings to support the main work objective:

- Understand the NDN architecture and its features. Survey the content privacy threats on in-network caching.

- Survey the side-channel timing attack and its countermeasure methods for content privacy.

- Understand the attack behavior and survey usable attack detection methods on related attacks.

- Develop an alternative attack design that is used to increase the success of the attack.

- Propose and develop a privacy model approach to preserve the cached contents while not compromising the distribution performance.

- Implement the scenario based on NDNtube and VoNDN applications to analyze the attack and DaD findings.

## 1.3 Research Methodology

To pursue the main objective of this work, the research methodology was identified by the list of objectives above. To achieve these, the NDN paradigm and its instruments were surveyed by bibliographic sources (IEEE, ACM, Springer, etc.). The literature allowed this work to understand the in-network caching content threats on privacy and their possible solutions in NDN. Therefore, this work focused on cache privacy-related works especially an attack-type called side-channel timing attack in NDN. The related works showed that the countermeasure methods can be a trade-off between privacy and cache efficiency respectively.

To overcome the statically configured countermeasures efficiency issues, the side-channel timing attack detection methods were presented to detect an adversary face to apply countermeasures. Therefore, the detection and defense (DaD) based algorithm was proposed. The DaD is used to detect the adversary's face and apply multiple countermeasures. Also, the different countermeasures are applied by the severity of the attack.

To meet with the work objective and algorithm implementation, the attack scenario implementations were scripted (C++, Python, SQL, and R) on the NDN simulator (*ndnSIM*) to evaluate the attack and DaD findings on NDNtube and VoNDN applications. In this implementation, an attack model called brute-force side-channel attack was developed to increase the attack success rate for multiple targets, compared to the traditional attack model.

## 1.4 Summary of Contributions and Publications

The conducted work of this research addresses the importance of side-channel timing attacks on NDN content privacy and proposed an efficient method to mitigate the attack for NDN applications. Considering the work objectives identified above, the contributions were summarized as follows:

1. *Survey traditional side-channel timing attack and propose a model that can be used to increase attack success.*

   The traditional attack design can be based on singular targets as described by recent researches. This may reduce the efficiency of attack considered for multiple targets and recently proposed attack designs. In this work, a brute-forced side-channel timing attack was developed to increase the attack success for multiple targets.

2. *Survey the countermeasures to mitigate the side-channel timing attacks in NDN. Study on attack detection methodologies in NDN.*

   The countermeasures can be effectively used to mitigate the attack. Also, these methods can reduce the content distribution which can be stated against the NDN paradigm. To reduce the countermeasure affects on distribution efficiency, the cache poising attack detection methods were investigated which can have similarities with a side-channel timing attack. Through the detected attack, possible detected adversary node

3. *Propose and develop Detection and Defense (DaD) approach that is used to distinguish the legitimate and adversary node then engage with the multiple the countermeasure methods.*

   The DaD is an approach to detect the attack first then the countermeasure method is applied. Additionally, the DaD uses multiple attack check periods to identify attack existence. If the attack has existed during these check periods, the DaD applies countermeasures starting with less effective countermeasures to more effective ones.

4. *Analyze brute-force attack results on NDN applications*

   The attack detection can be different on application and attack intention. For instance, the attack behavior is different on name and certificate attack. To differentiate an attack behavior, the brute-force is applied on streaming and VoNDN applications which are developed for this work as well. Therefore the DaD is proposed for streaming and VoNDN Instead of proposing a generic DaD for all NDN applications.

The following full-conference (Scopus-indexed) and journals (SCI-indexed) publications were produced to meet the objectives of this work.

**[journal-2] Content Privacy Preserving Approach in Video Streaming over Named Data Networks** (with *Costa A.* and *Macedo J.*), 2020. [preparation].

*Summary*. The streaming application (NDNtube) was developed and the brute-force attack was implemented in this work. The DaD algorithm was implemented and compared with the static countermeasure method on a large-set network topology set (AT&T).

**[journal-1] Public Key Certificate Privacy in VoNDN: Voice over Named Data Networks** (with *Gama O.*, *Costa A.*, and *Macedo J.*), IEEE, IEEE Access, 2020. [published]

*Summary*. In this work, the DaD algorithm was implemented to mitigate the attack. To do so, an NDN application called Voice over NDN (VoNDN) was developed to illustrate the attack. The brute-force attack results were presented which targeted the certificate privacy in trusted VoNDN application. To mitigate the attack, a static countermeasure method and DaD implemented. The DaD performance was evaluated and compared to the static countermeasure method.

**[conference-3] A Detection and Defense Approach for Content Privacy in Named Data Network** (with *Costa A.* and *Macedo J.*), IFIP-IEEE International Conference on New Tech., Mobility and Security, 24-26 June 2019, Canary Islands-Spain. [published]

*Summary*. The content retrieval time (CRT) and the attack success calculations were presented. The brute-force based side-channel timing attack was presented to increase the success of the attack which targets to name privacy. To mitigate this attack, the DaD algorithm firstly presented in this work, which detects the attack first then applies three countermeasures depending on the severity of the attack. The brute-force attack and countermeasures results were presented on large-set AT&T network topology using the NDNtube-like streaming application.

**[conference-2] Identifying Previously Requested Content by Side-Channel Timing Attack in NDN** (with *Costa A.* and *Macedo J.*), Springer, Future Network Systems and Security (FNSS), 2018-Book Chapter in Communications in Computer and Information Science. 9th-12th July, Paris-France. [published] [citation ≥2]

*Summary*. A side-channel timing attack scenario was implemented on tree (16 nodes) topology using the ndnSIM 2.5. The attack results were presented for the streaming application analyzed by cache hit ratio results. Also, a detection and countermeasure algorithm was presented to preserve privacy in NDN. The algorithm detects the attack based on RTT and cache hit ratio to apply random cache. Also, threshold value calculations for the RTT and cache hit ratio are presented to detect the adversary.

**[conference-1] Evaluating privacy attacks in Named Data Network** (with *Costa A.* and *Macedo J.*), IEEE, IEEE Symposium on Computers and Communications (ISCC) 2016-Full Paper, 27th-30th June, Messina-Italy. [published] [citation ≥9]

*Summary*. The Named data networking and privacy-related timing attacks are surveyed. Also, a detection method (cache hit ratio) was proposed to detect the adversary and apply a delay to the adversary.

## 1.5 Thesis Layout

The thesis is structured of seven chapters that were used to presenting the state of the art, the background of the research, the problem definition and its challenges, proposed approach, experimental setup, main results, and conclusions. Also, a summary of Chapters is presented by the following. Figure 1.1 illustrates the thesis working methodology.

Chapter— 2 *Named Data Networking*. This chapter firstly presents the IP network weakness for the content distribution and how it temporarily recovered from that weakness by discussing various caching server network designs. Then, this chapter introduces the NDN paradigm that attempts to remove the IP weakness by featuring in-network cache design while using different packet types. The chapter also summarizes NDN applications and content privacy threats especially on name, content, and signature (certificate).

Chapter— 3 *Side-Channel Timing Attack and Countermeasures*. This chapter discusses the side-channel timing attack and its countermeasure methods. This chapter also shows the privacy threats on the name, cache, and certificate in NDN. The side-channel timing attack success calculation and potential privacy risk in NDN applications are shown in this chapter. Lastly, the countermeasure methods proposed by other authors are discussed and pointed out the countermeasure methods efficiency affects the NDN cache-store.

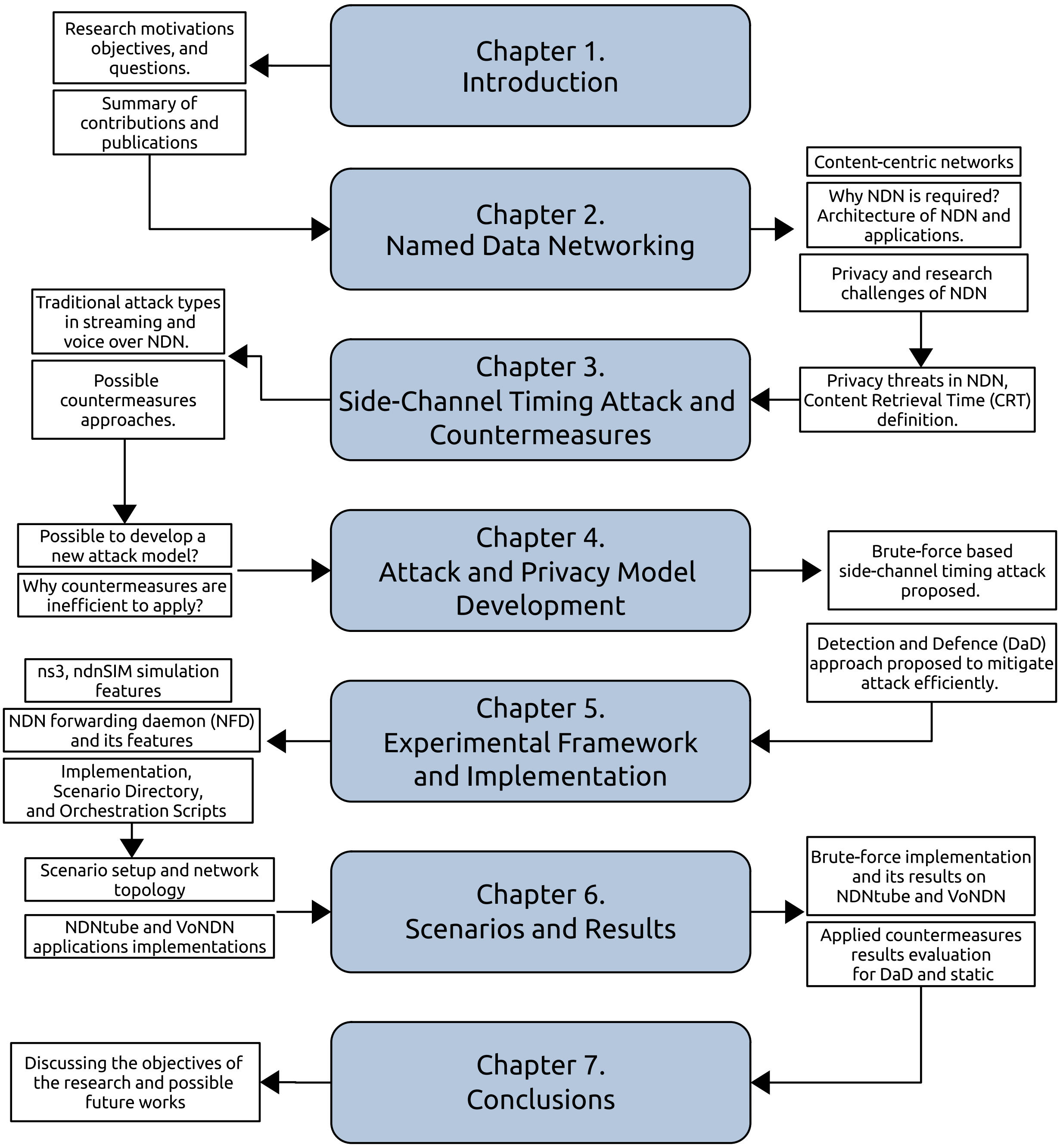

Research motivations objectives, and questions.
Summary of contributions and publications
Chapter 1. Introduction
Content-centric networks
Why NDN is required? Architecture of NDN and applications.
Chapter 2. Named Data Networking
Privacy and research challenges of NDN
Traditional attack types in streaming and voice over NDN.
Possible countermeasures approaches.
Chapter 3. Side-Channel Timing Attack and Countermeasures
Privacy threats in NDN, Content Retrieval Time (CRT) definition.
Possible to develop a new attack model?
Why countermeasures are inefficient to apply?
Chapter 4. Attack and Privacy Model Development
Brute-force based side-channel timing attack proposed.
Detection and Defence (DaD) approach proposed to mitigate attack efficiently.
ns3, ndnSIM simulation features
NDN forwarding daemon (NFD) and its features
Implementation, Scenario Directory, and Orchestration Scripts
Chapter 5. Experimental Framework and Implementation
Scenario setup and network topology
NDNtube and VoNDN applications implementations
Chapter 6. Scenarios and Results
Brute-force implementation and its results on NDNtube and VoNDN
Applied countermeasures results evaluation for DaD and static
Discussing the objectives of the research and possible future works
Chapter 7. Conclusions

Figure 1.1: Thesis Layout.

Chapter– 4 *Attack and Privacy Model Development*. In this chapter, the main contributions of this work are detailed. First, the brute-force side-channel timing attack design is introduced, then the Detection and Defense (DaD) algorithm is proposed to mitigate the side-channel timing attack in the NDN.

Chapter– 5 *Experimental Framework and Implementation*. This chapter mainly presents the network simulator 3 based NDN simulator (ndnSIM) features and objectives of the scenario implementation. Also, the NDN application layer protocol (NFD) features are described which is mostly used to implement the experimental objectives. Finally, the implemented components are detailed and discussed.

Chapter— 6 *Scenarios and Results*. In this chapter, the attack and countermeasure implementations were presented by elaborative results. To achieve the main objectives of this work, the adversary configuration and scenario topology setups are presented in this chapter. Also, the attack scenario findings and applied countermeasure methods are presented on small and large (real) network topology.

Chapter— 7 *Conclusions*. In this chapter, the main work contributions and its findings are discussed with providing the direction of future works. Also, the attack and countermeasures results were revised in this chapter.

# 2

# NAMED DATA NETWORKING

This chapter details the Named Data Networking concepts, comparing with the mechanisms in use currently on the Internet to deal with efficient global content distribution. The NDN architecture is presented, as well as all its components, with special attention to the routing and forwarding mechanisms based on names and contents. Finally, the NDN built-in security model is discussed and privacy issues are enumerated as research challenges.

## 2.1 Context

In the 80s the communication between computers brought global scale communication. Thus, the network and transport layers (IP and TCP protocols) were suggested for universal communication between computers. A set of procedures and rules defined by standard communication protocols (TCP/IP) were used to interconnect network devices over the Internet. Its communication network protocols were designed for point-to-point communications. The Internet idea was created to support people's communication and not multimedia content production and distribution. With the appearance and popularity of WWW, the Internet becomes a technology used to produce and consume real-time multi-cast multimedia such as online radio/video channels. However, the Internet was not designed for such a usage pattern, which brings a need for a real-time multi-cast design for today's Internet.

However, the usage of the communication world has changed dramatically since then. The growth of network devices, user expectations, the evolution of services, and ubiquitous interconnectivity are forcing the Internet to be overwhelmingly used for content distribution. Due to the TCP/IP paradigm network, the over usage of the Internet is creating network traffic congestion, server failures, and other unexpected network problems [2], [16].

The Information-Centric Networking (ICN) [22] is an emerging network approach that is an alternative to the IP. It aims to improve the scalability, cost reduction, network performance compared with the IP. Content-Centric Networking (CCN) [23] stands one of the approaches to ICNs. The CCN aims to content caching with various packet types, application layers, and protocol. Through these features, the CCN offers simpler and more potential options for alternate network paradigms.

NDN [16] is another ICN flavor, network paradigm alternative for TCP/IP networks. It is an ongoing project that proposes to transform the existing Internet design into content-centric architecture, to maximize the content [1] distribution efficiency. NDN is based on human-readable names and keeps the contents in the cache, thus facilitating content distribution and providing low latency [3].

## 2.2 Content Delivery Networks

The CDN networks [24], [25], [26] are built to support content distribution over the IP, an aspect of ICN architecture. The CDN is a collaborative collection of network segments into the Internet (IP), where the content(s) is replicated over the mirrored cache servers to perform effective and transparent delivery of contents to the end consumers. In CDN, the distribution services are about to replicate content(s) and cache the content(s) from the content provider to the distributed web servers [1].

The CDN name resolvers direct to the global Domain Name Servers (DNS), which can be used for load balancing. Briefly, the DNS resolves a domain name to IP addresses. For instance, consumer requests for a Uniform Resource Locator (URL), and DNS decides according to the hostname. Another benefit of the DNS service, they can take a role as a load balancer for CDNs as described by [26]. For instance, when DNS requests a certain domain name that is handled by a CDN, it determines the closest location to look-up that the DNS request to handle it. The DNS server does a geographic lookup based on the DNS resolver's IP address and then returns an IP address for an edge server that is physically closest to that area.

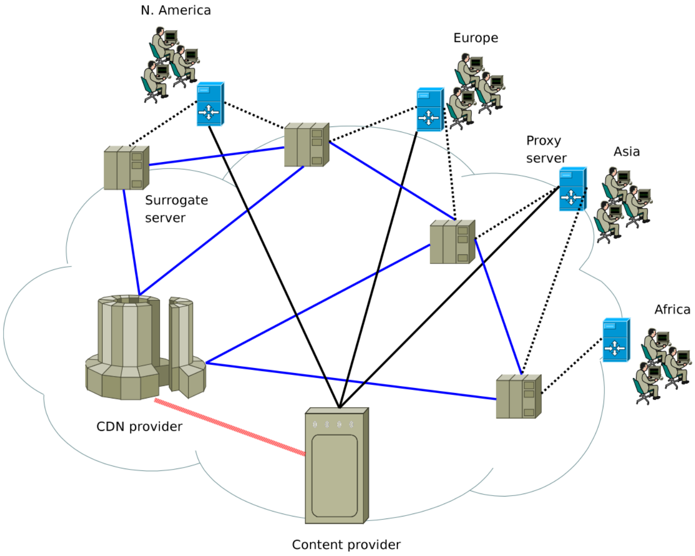


Figure 2.1: CDN Architecture (adapted from [1]).

[1] In this work, the terms "*data*" and "*content*" are used interchangeably.

The CDN has four basic components: CDN provider, surrogate, content provider, and end consumers. As Figure 2.1 illustrates, the content provider is the one who delegates the URL names of content(s) to be propagated to the CDN provider. Web caching is mainly implemented by proxy servers, whereas content replication is the main practice on CDNs. A company or proprietary organization (Akamai, Cloudflare, MaxCDN, etc.) can be a CDN provider. These organizations are providing infrastructure features to content producers to serve content in a timely and reliable.

## 2.3 Content-Centric Networking

The CCN (also called Data-Oriented Networks (DON)) architecture emphasizes the content directly reachable and routable. The CCN architecture can be adapted to IP and non-IP network architectures. The main objective of CCN is securing the content rather than securing the channel. Secured based content provides a flexible and scalable network. The CCN also identifies the contents by a name instead of IP address [27]. Through named and signed contents, their distribution can be done by caching servers such as CDNs as described by [28].

The CCN has mainly two packet types, The Interest Packet and Data packet. The Interest packet identifies the content that needed to be retrieved and is sent by the consumer. Then, the content publisher issues a content to be cached by nodes and retrieved by the consumer. Each of the interest packets is signed by the consumer and the content packets are signed by its producers, to maintain the integrity of the content.

CCN defines three types of entities: *i.* The Forwarding Interest base (FIB) is a table of name and corresponding of outgoing faces. The FIB is responsible to route the interest packets on longest prefix name matching. *ii.* The Pending Interest Table (PIT) is used to record faces by incoming and pending Interest packets. *iii.* The Content Store (CS) is used to cache the contents on caching servers or routers as described by [29].

The CCN is a form of ICNs. These concepts are known under different terms, including but not limited to: Network of Information (NetInf), Named Data Networking (NDN), and Publish/Subscribe Networking [22], [30].

In the next section, the Named Data Networking (NDN) [16] architecture is discussed in detail and NDN features are studied by comparing NDN with current TCP/IP networks.

## 2.4 Named Data Networking Architecture

The works of [16], [3] and [2] have proposed the NDN project, a network paradigm that is an evolution of the IP architecture. In NDN, any packet object can be named instead of naming the endpoints. This is a feature that changes the network semantics from packet delivery to the identified destination address to caching data by a given name.

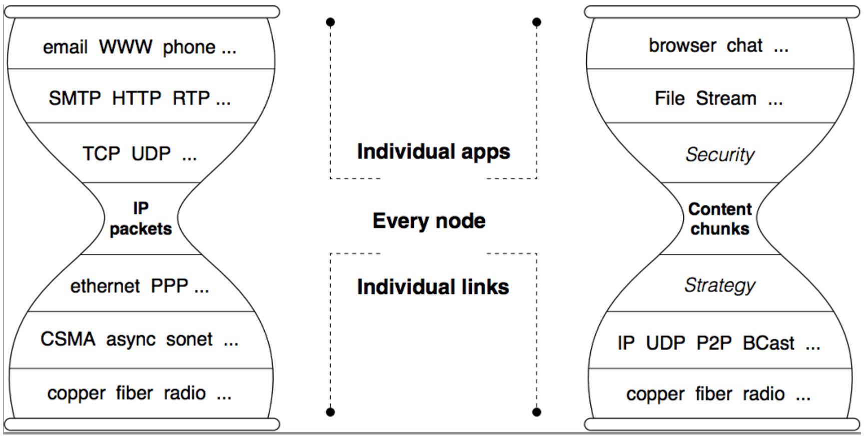


Figure 2.2: Internet vs. NDN hourglasses (adapted from [2]).

The hourglass model is used as a means of describing the Internet design. Today's Internet hourglass architecture represents a design in layered systems that aim to support a diversity of applications and implementations. The hourglass centers on the universal network layer (*i.e.*, IP). This thin waist is a key enabler of Internet growth, by letting the techniques of upper and lower layers to innovate independently, as described by [16]. In summary, the IP was designed to establish a communication network. However, the growth of social networks and applications has led to the use of the Internet as a content distribution network. Therefore, using the distribution networks via the communication network is error-prone and complex to solve. As Figure 2.2 illustrates, NDN keeps the same Internet hourglass-shaped architecture by replacing the thin waist with named data other than communication endpoints. This semantic changes the network from *delivering the packet to the given destination address* to a *caching data packet that is identified by a given name*.

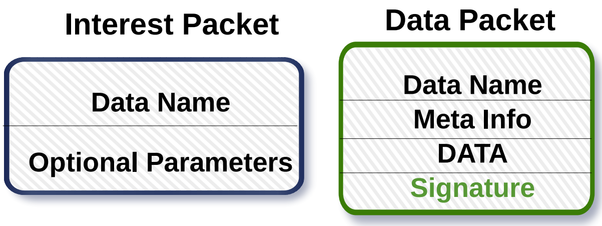


Figure 2.3: Packet types in NDN (adapted from [2]).

NDN is based on two main packet types: *interest* packets and *data*[2] packets. Interest packets are issued by the consumers and data packets by the producers. The content name in the interest packet identifies the request of the consumer, for example, */pt/uminho/algoritmi* is a name for a content, expressed in a

[2] In this thesis the terms "*data*" and "*content*" are used interchangeably.

structured way. It can be used to name contents related to the Algoritmi research center. As shown in Figure 2.3, the producer includes a signature in the data packet. Mechanisms for signing and verifying the integrity of the contents have been proposed for NDN, such as the one described in [16], [3].

In the following subsections, the four main NDN elements are discussed, the benefits of the design choices identified, and the challenges presented.

### 2.4.1 Negative Acknowledgment Packets

Technically, the NDN has two packet types of interest and data. However, the negative acknowledgment packet is another packet type it is used to identify the error reasons, when failures occur. The NACK is wrapped with one of error reason such as *e.g., duplicate, congestion, no route, denial of service, and time-out*, by the PIT, and sent to the FIB. In NDN, if the interest gets the existed content packets (retrieved), it is named as a *satisfying interest*, else the router gives up searching the data and the packet becomes a NACK by PIT and interest packet named as *unsatisfied interest* [31].

### 2.4.2 Names

The NDN name design is structured hierarchically, for instance, a video file may be produced by uminho <*uminho/vnetlab/intro.mpg*>. Similarly to the uniform resource locator (URL), with name components separated by '/' in a readable address format.

To request data by name, the consumer must build a name for the intended data. A name creation may be based on a deterministic algorithm, which lets the consumer and producer to gain information for the same name. A consumer may also retrieve contents by a partially known name. This technique is also known as "*longest prefix matching*" by interest selectors. They are used to precisely identify the content object. Retrieving data with a partially known name can be supported by a set of interest selectors.

Also, any NDN application and consumer is able to create their namespaces, to increase mapping data and its usage of the network. The naming of the data allows increasing mainly the functionality of data distribution, mobility, delay-tolerant networking, and multicast operations.

### 2.4.3 Data-Centric Security

To ensure the integrity of the data, each data packet is digitally signed by its producer. This is also known as NDN data-centric security, as discussed by [2]. It also supports data trust and allows the consumer to check the producer's public key validity. The validation can be also done by a hierarchical trust model, where some namespaces can be certified (certificate) by private companies or entities (third-parities).

The third parity of NDN application layers can manage the access control to data through encryption and distribution of the keys. Additionally, using the signatures on control messages and network packets allows securing the routing protocols.

### 2.4.4 In-Network Storage

NDN supports that any data packet can independently be retrieved from the network. Thus, the NDN router can cache data packets in CS (Content Store), to satisfy future incoming interests for contents. The CS is similar to today's buffer memories in IP routers. However, each NDN router can reuse a data packet while the IP routers cannot. Note that NDN handles the repositories (*e.g.* CS) and network channels as data retrieval sources.

On the other hand, the CS is a benefit for network congestion. In the presence of congestion, if it occurs for any reason, CS re-transmits the data. Imagine two congested links along a path between a consumer and a producer. If the requested data packet gets through the first congested link somehow, but couldn't get through the second one, then the data packet is dropped. But it remains in the CS of the intermediary node. Then, the consumer's interest becomes timed-out and the interest packet is resent. Caching will allow the data packet to be retransmitted to the consumer over only the second congested link. However, on the traditional Internet, the retransmission of the data packet can only be done by the content producer, and the data packet has to pass through the first congested link again.

So, CS presents optimal data delivery for static content, while getting supported by today's in-network repositories without having an application layer overlay. Even the dynamic contents, such as broadcasting or real-time conferencing, can benefit from CS in case of a packet loss.

### 2.4.5 Routing and Forwarding

In an NDN router, the forwarding of *interest* packets and *data* packets is carried out by three engines: Pending Interest Table (PIT), Forwarding Information Base (FIB), and Content Store (CS) [3].

CS represents a cache for data packets, similar to Internet routers buffer memory. FIB is a name prefixes routing table and respective outgoing interfaces, used to route interests. PIT is a pending interest table and a set of corresponding incoming interfaces.

As Figure 2.4 illustrates, when a consumer interest is received, a lookup is made on the CS for previously cached content that matches the name requested. If there is a matching content, it is sent as a reply with a data packet. If the data packet is not in the CS, the router checks the data name in the PIT. If there is a matching name in the PIT, the router records the incoming face of the interest for the future reply and stops the procedure. If not, a new PIT entry is created for that name, recording the incoming interface, and a lookup is done on FIB. The interest is routed using FIB information to the producer. For each forwarding

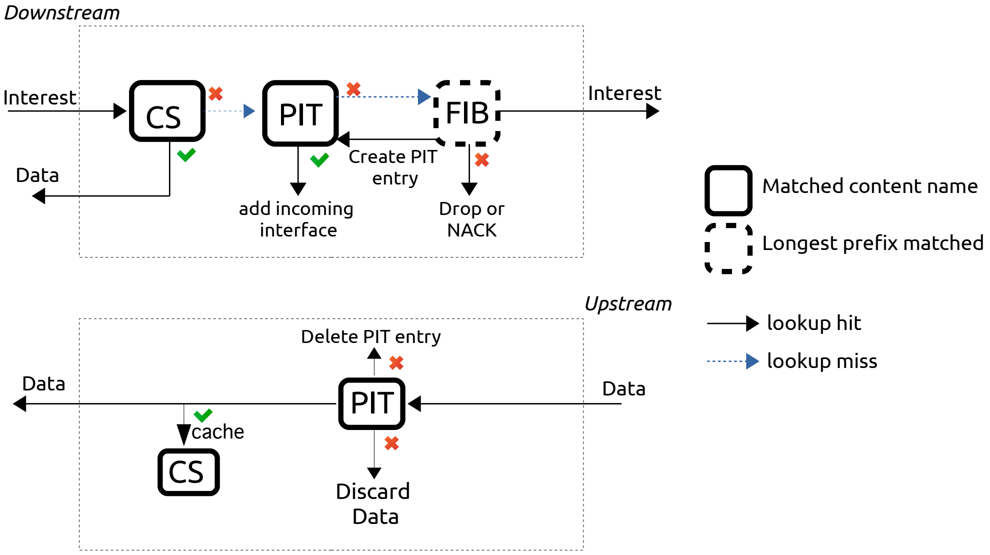


Figure 2.4: NDN forwarding engine model (adapted from [3]).

interest packet, the longest name prefix match is a lookup in the FIB, which determines where to send it. The list of outgoing faces of the FIB matched entry is an important reference for the routing. In case a name is not found in FIB, the interest becomes unsatisfied. When all FIB lookup misses are replied with a NACK packet, the forwarder can limit the requests [16], [3]. For example, This can be used to mitigate the denial of service attack in NDN, as described in [32].

When a data packet arrives at the downstream router, the PIT of this router is checked for a matching entry. If a match is found, the data packet is forwarded and stored to the CS, and the PIT removes the entry. If the signature verification of the data packet fails, the PIT discards it.

### 2.4.6 Table Management

The NDN forwarding path contains three tables: FIB, PIT, and CS. Also, these tables can be managed by special NDN applications (NFD, NLSR, and RIB [33], [33]). These can be used, for example, routing for routing information to each other to populate the tables.

As Figure 2.5 illustrates, the FIB is used to forward the interests to the potential sources. The FIB is updated by FIB management protocol, which is operated by the routing a forwarding application layer.

Table 2.1: Forwarding Information Base Process.

| **Name prefixes** | **Stale Time** | **Interfaces ranked by forwarding Policies** | | | | |
|---|---|---|---|---|---|---|
| Name prefix | Time | Interface ID | Routing preference | RTT | Packet status | Rate limit |

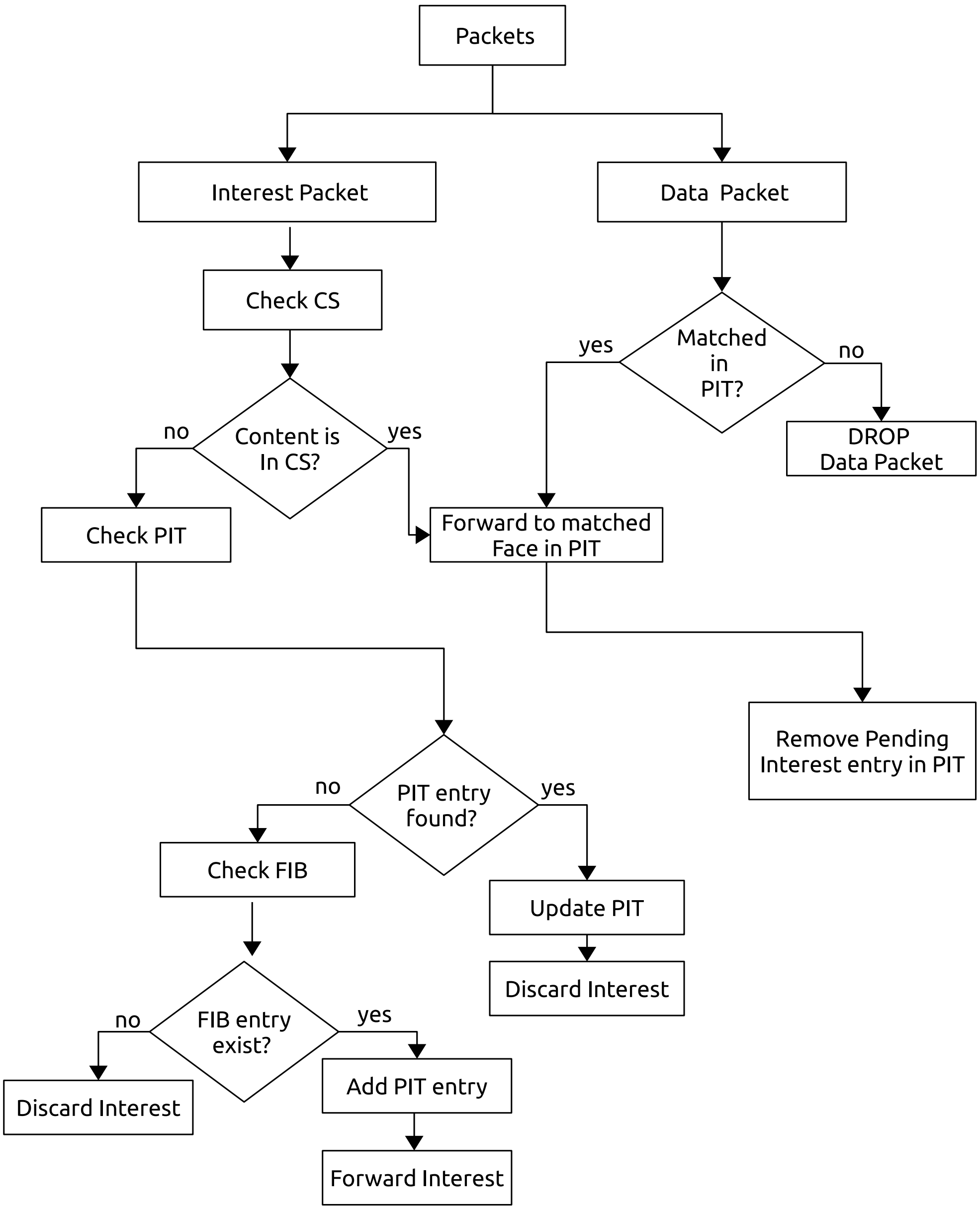


Figure 2.5: Single Interest/Data packet forwarding mechanism (adapted from [2]).

As Table 2.1 shows, the FIB manager processes these take the orders from the routing information base (RIB), that receives the routing pieces of information (static or dynamic) from the NDN application daemon (*e.g.* NFD) [34], [33].

The CS is searched before the incoming Interest is given to the forwarding strategy for further processing. This way, the cached data, if available, can be used to satisfy the Interest without actually forwarding the Interest anywhere else.

CS performance has an impact on the performance of the NDN network because it stores a large number of packets, and virtually every request packet accesses the CS. The choice of the underlying data structure for an efficient lookup, insertion, and deletion, and the cache replacement algorithm (*e.g.*, FIFO, LRU, LFU) are crucial for maximizing practical benefits of in-network caching.

The PIT tracks each Interest packets that are forwarded upstream toward the content producer. Table 2.2 shows that it also records recently satisfied interests for several measurements (namely send-time, output, and input used interfaces, and interest lifetime) and loop detection (nonce list) purposes.

Table 2.2: Pending Interest Table Process.

| **Names** | **List of Nonce** | **List of ongoing interfaces** | | **List of ongoing interfaces** | |
|---|---|---|---|---|---|
| Content name | Nonce | Interface ID | Lifetime | Interface ID | Send-time |

### 2.4.7 NDN Transport Function

The Internet was developed to manage point-to-point communication functions. The transport layer is one of the layers on the Internet stack. It provides communication between applications within layered protocols such as *i.* User datagram protocol (UDP), *ii.* Transmission Control Protocol (TCP). The transport layer may also provide the management of error corrections such, as ACK and NACK.

In NDN, there is no transport layer. The transport functions are not in a separate layer and may not be separated from other layers, like on the Internet. Today's transport functions like ordered reliable delivery, demultiplexing, congestion control, are part of the forwarding layer or application layer. This can be done using the names at the NDN stack by applications. For instance, the trust scheme, and application process reliability controls.

Any NDN router can manage the traffic load by controlling PIT size on a hop-by-hop base. For example, if a router is overloaded by incoming traffic, it simply resizes PIT and CS or stops sending interest to the next NDN hop. This feature illustrates that NDN may eliminate dependency on end consumers' traffic congestion control, as proposed by [16].

NDN C++ library with eXperimental eXtensions (ndn-cxx) can implement the NDN primitives. These can be used to write various NDN applications such as NFD, NLSR, NAC, NDNS, etc. Application-driven development also increases the functionality of certificate verification. NDN application research maintains five key areas: *i.* trust models, *ii.* namespaces, *iii.* in-network storage, *iv.* packet sync and, *v.* bootstrapping and discovery.

The reliable content delivery can be done by the application or supporting libraries which may monitor the status of outstanding Interest packets and retransmit them (*e.g.* after a timeout). Also, each NDN router has a flow balance mechanism available to control its network traffic, by limiting the PIT at each hop. This brings effective network congestion control to the NDN. For instance, in case of network congestion occurred, it may be mitigated by CS. Thus, NDN can avoid the kind of congestion collapse that can occur in today's Internet when a packet is lost near its destination and repeated retransmissions from the source host(s) consume most of the bandwidth.

## 2.5 Security

In NDN, security is a built-in data function, rather than being a function of how and where data is obtained. The producer signs the data packet binding its name to data. In NDN, it is mandatory to use signatures. So, each data packet and corresponding application must meet security requirements.

The signature allows the determination of the data producer, letting the consumer's trust into a data packet to be de-bound from its origin. NDN supports a fine-grained trust scheme. This allows determining whether a public key owner is an acceptable content publisher, for a specific piece of the data, into a particular context.

### 2.5.1 Data-Centric Authentication

The data producer generates the digital signatures used to authenticate the data packets. The signature binds the producer name with the data packet. To authenticate the signature, a consumer can verify the digital signature using the producer public key certificate, regardless of the data packet location.

A consumer may also bind the publisher's public key of the requested interest packet that is called PublisherPublicKeyDigest (PPKD). With PPKD within the interest packet, a matching data must have the same digest in its PPKD, to validate the data publisher. This match-up is also known as the interest-key binding (IKB) rule, as described in [35].

A data producer binds the name and digital signature key of a data packet together, to indicate the certificate name, which is called a KeyLocator. It is used to retrieve the publisher's public key and to determine the legitimate data name by the consumers.

The consumer simply verifies the identity name of the content producer, which is associated with the producer public key into a certificate name. Therefore, the consumer or the router can discard directly data, if the certificate does not provide correct identification of the producer.

Since the PIT can hold each interest packet that is sent and waiting for content as a response, it can be used to verify the content publisher's signature from previously cached requests.

To trust the content publisher's public key, a consumer may ask for the validity of the public key included in a certificate. This may occur in another operation, called a chain of trust. The process requires getting multiple public-key certificates, that must be validated until reaching a valid root signature. The verification is done using the chain of trust. A root certificate can be the trust anchor, from which is derived all chain of trust. When is digitally established the trust chain, the consumer can authenticate the signature using the public key certificate.

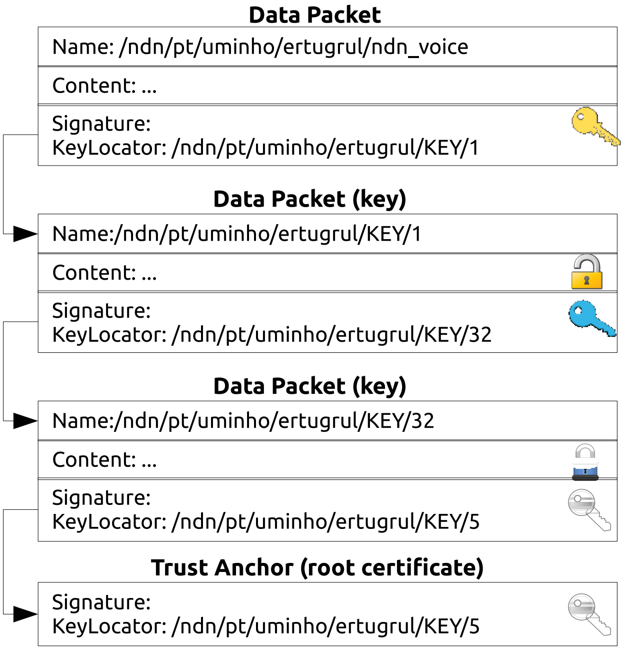

Data Packet
Name: /ndn/pt/uminho/ertugrul/ndn_voice
Content: ...
Signature:
KeyLocator: /ndn/pt/uminho/ertugrul/KEY/1
Data Packet (key)
Name:/ndn/pt/uminho/ertugrul/KEY/1
Content: ...
Signature:
KeyLocator: /ndn/pt/uminho/ertugrul/KEY/32
Data Packet (key)
Name:/ndn/pt/uminho/ertugrul/KEY/32
Content: ...
Signature:
KeyLocator: /ndn/pt/uminho/ertugrul/KEY/5
Trust Anchor (root certificate)
Signature:
KeyLocator: /ndn/pt/uminho/ertugrul/KEY/5

Figure 2.6: Data packet authentication in NDN (adapted from [4]).

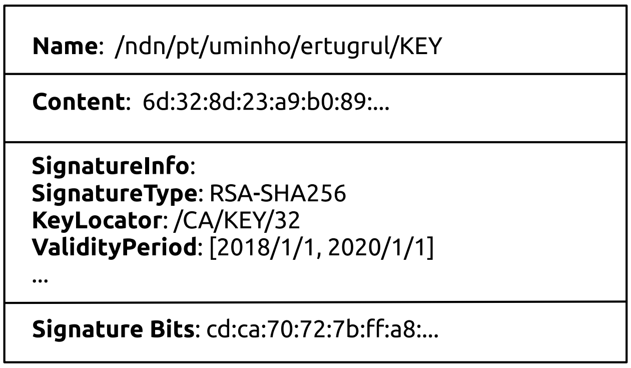
The trust must be established to eliminate the fake (un-legit) content producer(s) that may produce fake data with a legitimate data name. In theory, a trust chain example is illustrated in Fig. 2.6, between the content and the signer, between the signer and the keys of the administrators, between administrators' keys and other administrators' keys, and finally between administrators' keys and the configuration key.

### 2.5.2 NDN Certificate

Similarly to any data packet in NDN, a certificate bounds a public key with the name through a signature. If the identifier is a data name, the data packet containing public key bits is considered effectively as a certificate [36].

Figure 2.7 illustrates the NDN certificate. In NDN, a certificate can be presented as like any other content that is carrying a public key. The “key” refers to a content packet that carries a public key. The KeyLocator refers to the certificate issuer or certificate authority.

A data packet seals the binding between name and data through a digital signature. The advantage of using the certificate as a data packet is that a consumer can retrieve and validate a certificate by issuing an interest packet. Indeed, the public key certificate has the general format of a data packet. For instance, a producer expresses a certificate challenge using name and content to carry the public key bits. Next, there is a discussion about why X.509 [37] is not suitable for the NDN certificate.



**Name**: /ndn/pt/uminho/ertugrul/KEY

**Content**: 6d:32:8d:23:a9:b0:89:...

**SignatureInfo**:
**SignatureType**: RSA-SHA256
**KeyLocator**: /CA/KEY/32
**ValidityPeriod**: [2018/1/1, 2020/1/1]
...

**Signature Bits**: cd:ca:70:72:7b:ff:a8:...

Figure 2.7: NDN Certificate Format (adapted from [5]).

**The unsuitability of X.509 for NDN Certificates.** The X.509 standard defines the traditional point-to-point Internet certificate format. However, X.509 cannot be adapted for NDN (except the public key bits), due to the format differences between the two certificates. Firstly, the NDN trust requires a relationship between the name and the key name. On the other hand, the X.509 has only its naming system, which makes it inadequate for its operations with an NDN certificate. For instance, NDN can have a strict hierarchical naming system. In the NDN mechanism, a name converting can be also used between multiple naming systems. However, an X.509 may not have a strict hierarchical structure to convert names because an X.509 certificate name is a string (*e.g.* Google Internet Authority G2) [38], [5]. Secondly, X.509 requires auxiliary dependencies, such as the Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP), built over IP [4]. The retrieval of X.509 certificates is only feasible over an established point-to-point channel.

For all these reasons, NDN requires an own certificate design for the operational challenges described before. As a summary:

- NDN is based on a naming scheme and requires trust between a name and key name.
- The NDN certificate must be retrieved independently from the network.
- NDN certificate has no dependency on IP.

**Certificate Requirements.** A certificate may have the same requirements of a data packet and can be retrieved from the CS, thus allowing the validation procedure for both keys and data. However, the data packet form does not include support for specific requirements for the certificates, such as additional information. The certification structure requires an extension of the data packet format.

The requirements are classified by Naming Convention, MetaInfo, Content, and SignatureInfo [38], [5], as presented following:

**Naming Convention.** The generation of a unique certification name requires a name convention for the name components, as shown in Figure 2.8. The name is divided into several components with different semantic. These are Subject, KeyID, IssuerID, and Version.

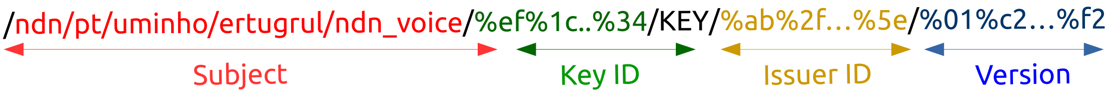

/ndn/pt/uminho/ertugrul/ndn_voice/%ef%1c..%34/KEY/%ab%2f...%5e/%01%c2...%f2
Subject
Key ID
Issuer ID
Version

Figure 2.8: NDN Certificate Naming.

- **Subject**. A name for data packet.
- **KeyID**. A name is used to identify a key that is bind with the subject. It uses as the crypto hash function of the public key, such as SHA-256, the same as X.509.
- **IssuerID**. A name that distinguishes the certificate by multiple certificate authority (CA).
- **Version**. A name is used to distinguish the certificates of the same subject name by the same issuer.

**MetaInfo.** The MetaInfo consists of additional information for the data packet, namely the ContentType, FinalBlockId, and FreshnessPeriod. Since the data packet has public key bits, ContentType is used to define the content as a certificate key. When the content cannot fit in a single data packet, the FinalBlockID is used to identify the certificates in more than one data segments. The optional FinalBlockId identifies the final block in a sequence of fragments. It should be present in all fragments to provide advanced warning of data packet fragmentation. The FreshnessPeriod is defined by the producer to indicate how long a router should cache the certificate before being discarded. Note that, the FreshnessPeriod should not be confused with the ValityPeriod for certificates.

**Content.** The certificate content has a hash public key format (SHA-256). This encoded information is the same as the X.509 public key format. It should be noted that this crypto hash function can be changed accordingly to the network needs.

**SignatureInfo.** The identification as a statement of certificate issuers and its attributes are in SignatureInfo. SignatureType and KeyLocator are the classifications of statements. The SignatureType and KeyLocator specifies the type of signature (*e.g.* RSA with SHA-256) and key name (*e.g.* */ndn/pt/minho/ertugrul/KEY/32*). However, the lengths of the statements are not sufficient to accommodate more attributes. To accommodate more attributes for extensible authentication, Type-Length-Value (TLV) blocks extend SignatureInfo. These are consumer-driven attributes which categories are non-critical and critical.

If verification fails in a critical attribute certificate, the certificate becomes invalid. The consumer can also drive a non-critical attribute. Two certificate attributes are defined following:

- ValidityPeriod. This is a critical attribute for a certificate and a restriction for the lifetime of a signature. The classifications of timestamps of the certificate validity are the start time (NotBefore) and expiration timestamp (NotAfter) of the certificate validity.

- AdditionalDescription. Its classification is a non-critical attribute and it describes additional information, such as a set of key-value pairs. The key-value pair can be optionally identified by the issuer, to keep the maximum integrity of a certificate.

**Certificate Request Procedure.** The owner of the key may request a certificate, to verify its key. The first procedure is the selection, by the owner, of an issuer profile, loaded into the certificate agent (*e.g. /ndn*). Then, the agent of the certificate may ask for additional information (*e.g.*, the email address of the NDN-testbed, for instance) to the owner of the key. With the additional information, the certificate agent follows the issuer profile rules, to convert an email address (*e.g. ertugrul@vnetlab.gcom.di.uminho.pt*) to an NDN name (*e.g. /ndn/pt/minho/ertugrul*). Then, the certificate agent builds the key name (*e.g. /ndn/pt/minho/ertugrul/43/KEY*) by knowing the requesting public key.

**Interest packet structure for a certificate request.** The certificate request is expressed in the Interest name, using encoding information. As shown in Figure 2.9, the Interest name of a certificate request has six components.

/[ServicePrefix]/Request/[KeyName]/[KeyBits]/ChallengeSelection/[AgentKeyBits]

Figure 2.9: A Interest packet for certificate request (adapted from [5]).

- ServicePrefix. A service that provides where an Interest packet needs to be forwarded.

- Request. Identifies a request name for the certificate.

- KeyName. A public-key name.

- KeyBits. Public-key bits.

- ChallengeSelection. Additional information section and needed to be answered to the certificate requester.

- AgentKeyBits. An agent key to keeping maintenance.

### 2.5.3 Self-Certifying Content Name

Self-certifying content names to names (SCN) for contents, where the name itself is cryptographically structured. This can be used to determine the integrity of the producer. The simplest form of self-certification, hash-verified data, simply names a piece of content directly by its cryptographic digest (ex: SHA-1) [39].

The SCN can be also used to mitigate an attack called cache poisoning in NDN. Maliciously constructed contents may fill the content store, to affect its cache structure. This adversarial behavior is called cache poisoning. The SCN allows the routers to identify whether a given content is a "legitimate answer" for a specific interest packet. Each name contains two parts: *i.* publisher's public key, and *ii.* object identifier. More specifically, the work of [35] proposed Interest-Key Binding (IKB) rules for SCNs. The IKB rule of each interest must reflect the public key of the producer.

### 2.5.4 NDN Trust Management Applications

In this section, we evaluate the trust management (schema) applications and their features in NDN.

**Application layer Trust Model.** The NDN application layer may take responsibility for the trust. The trust model defines the processes used to certify public keys using certificates and validate packet signatures. The application layer trust model defines the requirements of identity certificates. The application layer trust model is used to validate command interests. Command interests are digitally signed interest packets. They are crucial in many contexts. For instance in routing contexts. The link-state routing module may require a trust model to sign routing messages [40]. This trust management can be done by NDN routing application (*e.g.* NLSR) or other trust management applications. The command types, such as faces, FIB, and strategy-choice, are sent to the Forwarder engine while routing commands are sent to the routing manager ([34]). In this application, the layer trust model uses command interests, and a routing manager trust model, is next presented. In the following sections, a set of specific application trust models are presented, like command interest, routing trust model, building management trust model, video/voice trust model, vehicular and other examples.

**Command Interest.** The command interest is used to issue authenticated control commands. The command interest has five components after the management name-space, and it looks like this: */signed/interest/name/<timestamp>/<nonce>/<signatureInfo>/<signatureValue>*. The command interest will be invalid under the following four conditions:

1. If the interest has one or more missing components (signatureValue, signatureInfo, nonce, and timestamp),
2. If the signature fails with the public-key by SignatureInfo,
3. If the key is not trusted for signing the control command by the responsible trust model,
4. If the content producer has already received a valid signed Interest.

**Routing Manager Trust Model.** The routing manager may rely on its trust model to validate routing type command interests. The manager may identify the conditions for keys to be trusted to sign routing commands. The corresponding trust model must be able to answer these questions:

1. How to authenticate signers?

2. Who are the trusted signers for routing command interests?

The name of the key with an NDN regular name expression defines the trusted signers [41]. If the regular expression does not match the signing key, the command interest will be invalid. The signers must follow the rules that manage how a signing key is validated through a trust chain back to a trust anchor. The authentication and identification of the signers follow the NDN Validator Configuration File Format specification [41].

**Link State Routing Trust Model.** Currently, NDN uses a featured routing protocol called NDN-based link-state routing [40]. It uses names to identify data, networks, processes, routers, and keys. Link state routing is also using underlying communication channels, such as Ethernet, TCP/UDP/IP.

The link-state routing application protocols may use their trust model or other NDN trust applications to distribute keys and do derivative trust operations. From the trusted application, an application can fetch and verify the data signature through the *KeyLocators*. However, the authenticity of the fetched key is challenging. To overcome the authenticity errors, the link-state routing requires a simple hierarchical trust model that proposes a trust relationship for bound names and keys. For example, a root key belongs to an administrator, and the under-layer of the root key holds the site keys, which are owned by a site administrator. The site keys sign the operator keys, which in turn sign the router keys and the link-state routing process of the NDN router. Lastly, the key signs of the routing content originated by link-state. This matched name trust delegation can be represented as "*/root/site/operator/router/process*", as described by ([3]).

**Building Management System Trust Model.** Since NDN separates the namespace from the trust model, it becomes an ideal network design for Building Management Systems (BMS) as firstly proposed by [3]. More specifically, the BMS design uses trust by control sensors proposed and publishes the content in three namespaces [42]:

1. A namespace for application data access for physical building operations;

2. A namespace for bootstrapping and device discovery;

3. A namespace for trust management for keys that identifies the faculty roles and relationships of the principal, i.e. the building management.

The authorization of the keys may limit its scope in the trust hierarchy. For example, the hierarchy authorizes the keys to sign a particular name. To overcome this limit, a BMS is proposed by [42]. This BMS is a hardware tool that separates a hierarchical namespace for content, encryption-based access controls, and proprietary protocols.

**Video/Voice Streaming Trust Model.** A simple multimedia control system, video conferencing, teleconferencing, and live-media broadcasting can be supported by all NDN applications that are receiver-driven. Since NDN does not require a centralized server, it supports multi-user peer-to-peer voice/video application protocols, such as VoCCN (Voice-over CCN) proposed by [7], Real-Time Streaming Data (RTSD) [43], and Real-Time data Retrieval (RDR) [44] for the RTSD applications.

The trust may vary according to the application protocol requirements. For instance, to verify the trust in per-packet on its signature, the RTSD uses trust schemes, such as a model called schematizing trust in NDN, as described by [4]. Another serverless NDN chat application is called ChronoChat, as described by [45]. The application is supported by a non-hierarchical trust model and presents an encryption-based access control.

**Vehicular Network Trust Model.** NDN supports vehicular networks for location-based content retrieval and trust model proposals. The vehicular applications also include updates in the NDN architecture to include 3G/LTE, WiFi, WiMAX, and DSRC/WAVE [3]. The router can cache a broadcasted certificate to provide secure data to other vehicles or send it to the corresponding certificate authorities to control the certificate integrity [46].

**Usable Trust Model.** In theory, the trust chain must have to collaborate with the chain members. However, in practice, if any chain member configuration(s) setting mismatch (expired or issuer name) to the next chain member, in this case, the trust chain may end up with non-trivial or error-inclined. A weak trust implementation error may compromise the sensitive private keys. For example, if a producer associates the consumer management with the consumer keys rather than with the administrator keys, it can allow one consumer to authorize another consumer without the authorized administrator's permission. To tackle this problem, the trust relationships must be captured by a set of trust schema agent called schematizing trust in NDN, as proposed by [4]. A system-level agent can interpret well-defined rules and execute the signing and authentication procedures, to establish integrity. The proposed trust schema agent is based on a notation for the authentication and signaling process. Such notation can be the usual NDN name pattern.

**Next Generation NDN Repository.** The repository (repo) is a larger version of CS in NDN. The repo can exist in any NDN node, to preserve data. The repo protocol uses the NDN client library that is used to reading, insert, and deletion of data objects.

The repo semantics is based on signed interests that includes <SignatureInfo> and <SignatureValue> as the implementation identified by [47].

## 2.6 Privacy

The digitally signed packets do not guarantee to protection against traffic analyzes as described by [48]. Since each NDN node must have cache content, the privacy concerns increase for cached content. Each cached content may be targeted by an adversary in NDN. The recently cached content characteristics may be different than uncached ones, the adversary uses this information by determining the cached contents from CS.

When a consumer retrieves the data packet, the network infrastructure may keep information about the requested content, to improve the throughput and latency of the network. The information may present publicly the consumer, the content producer, the public key certificate, the name, and the content size. The adversary may reveal this information to determine cached content from CS.

Suppose the adversary (*Adv.*) wants to determine whether a consumer (*Bob*) has recently requested a data packet (*C*) or not. If the Adv. share the first-hop router (*R*) with Bob, the Adv. measures the estimated RTT (round trip time) [$R \longleftrightarrow Adversary$]. Then, it issues interest of the content (*C*) and compares these two RTT values, to determine if $C$ has been recently cached by $R$[3]. Also, the Adv. may determine whether the producer (Alice) has been recently asked for the content (*C*) or not.

Similarly, suppose that Alice and Adv. share the same router or separated at least by one router. The Adv. estimates RTT [$Alice \longleftrightarrow Adv.$] and then issues an interest requesting for $C$. The Adv. concludes that at least one consumer has requested $C$ and cached by the router(s) if this RTT is lower than the former RTT. Lastly, the Adv. may combine these two attack types against Bob and Alice to determine if they have exchanged packets recently or they exchange packets in real-time two-way communication, *e.g.* SSH or voice/video [17][4].

Since the interaction between the Adv. and the router is natural, the two attack types are simple and do not require the Adv. to have any privileges. Such attack types, named ”timing attacks”, allow the Adv. to learn whether a neighbor consumer recently requested a certain content. The timing attacks and its countermeasures are studied in the next Chapter 3.

### 2.6.1 Internet vs. NDN Privacy

Due to the architectural differences between the IP and NDN, the privacy issues may be different from each other. The IP communication networks may offer weaker privacy than NDN. For instance, an adversary node may find out what is in the IP packet by checking the payload, or who has requested the content packet by inspecting the destination, and address, as described in [20].

However, NDN only is interested in what data is being requested and not who is requesting it, and routers cannot have any information regarding the consumers. Ideally, even if the router has been compromised, it may facilitate observation of what data packet is being requested without identifying who is requesting it (one is connected to the same host) as discussed by [16].

[3] We do suppose Bob and Adv. only connected to the router. If there are other users, this will not change the nature of the attack

[4] In the real-time voice/video attack, the Adv. must be physically neighbor to the target (*e.g.*, same shared Ethernet interface)

Table 2.3: Information privacy considerations on Internet, CDN and CCN/NDN.

| Privacy Considerations | Base Internet Model | CDN | CCN/NDN |
|---|---|---|---|
| Source Address | End-points identifiable | End-points may be identifiable | Ideally, end-points are not identifiable due to lack of source address |
| Destination Address | Destination addresses identifiable | Destination addresses may be identifiable | Destination addresses are not identifiable due to lack of destination address |
| Round Trip-Time | Identifiable between source-destination | Identifiable between edge server to destination | Identifiable between node to destination |
| Content Name | Indicates content producer | May be indicate content producer | Due to lack of source address, content name does not indicates producer |
| Packet Size | Identifiable | Identifiable at edge cache server | Identifiable at each cache node |
| Cache | Does not have cache | Only represented at edge cache server | Each node cache is identifiable |
| Certificate | Public Key Certificates may indicate content producer (if certificate exist) | Public key Certificates may indicate content producer at edge cache server | Public key Certificates mayindicate content producer at each node |

Table 2.3 compares the base IP and CCN/NDN privacy considerations in their caching server/router. The privacy survey analysis from [48], [49], [50], [51], [52] have classified the privacy considerations under seven main subjects.

**Source Address**. The source is used to identify the content producer on the Internet and NDN. If the communication established point-to-point the content source address may be used to illustrate its producer such as in the base-Internet model. If the communication is based on content-centric such as CCN/NDN, the content may be retrieved by several cache nodes, making it hard to define who produced the content. Therefore, every-node caching networks such as CCN/NDN are privacy-friendly due to lack of source address as described by [51].

**Destination Address**. Similarly, like an identifiable source address, the destination address may be identifiable in the Internet base model, the packets are carrying the destination address, which may affect consumer privacy. However, if the networks are based on content (content-centric), the content does not have a destination address because they can be answered by any caching based nodes as described by [52].

**Round-Trip Time**. The RTT is a time measurement value between issuing a content request and obtaining its answer. The caching networks aim to achieve RTT value close to 0, to improve content distribution efficiently. However, the attack may take advantage of this value, to identify cached/un-cached contents. This may affect content privacy.

**Content Name**. The content name is used to identify the contents. In the base Internet model, the IP name identifies the content producer. In case of CDNs may illustrate the source-destination address. If CDNs are content-centric, it's a challenge to identify where the content is produced. However, in the caching networks (CCN/NDN), the contents may be cached by several nodes, which does not identify where the was content retrieved from.

**Packet Size**. Whether the content is encrypted or not, the content size can be identifiable in base Internet and caching networks.

**Cache**. Since the base Internet model does not have a cache [5], the cache privacy does not matter. In caching networks, the adversary may able to identify cache size and content by multiple requests.

**Certificate**. If the base Internet model has a trust model, to build-in the integrity of contents, the producer public keys may be used to identify the producer. This issue is the same in caching networks. Indeed, the cached certificates are used to increase the content distribution, therefore the cached certificates are used to identify producer and consumer in real-time communications.

NDN architecture may promote some important privacy issues, mainly caused by the semantic richness of the content names as pointed out by [16]. The main privacy issues are studied in the next subsections.

### 2.6.2 Cache Privacy

The NDN contents may be cached within defined cache policies, therefore, the node's cache presents how many contents have been cached and what is the size of the node's cache. Indeed, the number of cached contents may illustrate the number of consumers who are requesting the contents from the cache.

[5] During the transmission, the cables can carry the contents for a while, that considered as a temporary cache, we are ignoring that.

In this case, the adversary takes advantage of RTT replies by timing attack, to identify cached contents number and the size of the node's cache. For instance, if the adversary is sharing the first hop with the target (consumer), the privacy risk is maximized due to the limited number of consumers sharing that cache. If not, the adversary still may determine away hop's cache, by analyzing multiple RTT replies.

### 2.6.3 Content Privacy

In IP, the channel may be secured between the end-points. That limits the adversary, to inspect the source/destination address, type of service, and additional packet information.

However, NDN packets are individually secured by content-centric design and cached in the node. That advantage the adversary effectively inspects the content of each node whether they are secured or not. Ideally, the CCN/NDN contents provide strong privacy because of a lack of source/destination address. However, the contents are available for any consumer that ask for it, the adversary can monitor cached contents for censorship purposes as described by [53].

### 2.6.4 Name Privacy

Currently, the HTTP header names (IP to URL name, metadata, content-type, content length, etc.) may reveal information about the packet itself. To overcome this issue, the header names can be encrypted by HTTPS (secure HTTP) connection.

However, the NDN router routes data packets based on names and each human-readable name is related to the content itself. Such a human-readable name design may have an increased level of threat to the content names than IP URLs. Since, we cannot use a secured channel in NDN because its content-centric, the work [21] suggested encrypted names, to increase the level of name privacy in NDN. However, the content names must have to be visible for NDN, the encrypted names are not an ideal solution for content names. Even if the content name is not human-readable, the content name can be still threatened by an adversary because the name of the content may require an NDN Domain Name Server (NDNS) that is proposed by [54], which translates the encrypted names to human-readable.

### 2.6.5 Signature Privacy

All NDN contents are digitally signed by its producer, to provide integrity and guarantee on provenance which makes all signatures publicly verifiable by the NDN nodes and application layers [6].

[6] Currently the NDN offers RSA and ECDSA signature algorithms.

However, digitally signed contents are may leak sensitive information about the content signer. Because contents carry a public key that is publicly fetchable by any consumer, the adversary may be able to determine content producer by using a timing attack.

For instance, the two-way conversation tool VoCCN [7] is based on signed contents, to keep content integrity between the callee and caller[7]. In this structure, the certificates may be used to verify the content producer. Since the certificates are effectively a content and cached in the NDN node by the application layer, the adversary may use the timing differences, to identify the call history by timing attack.

## 2.7 Why Named Data Network?

NDN is a CCN's academic dual. However, NDN does not offer IP adaptation at all, but it may use existing CCN components such as packets, name, and caching nodes. The NDN was proposed to overcome IP operational limitations. The improvements can be categorized by processing, failure handling, transmission, control, and storage (buffer-memory) as presented by the works [52] and [15].

Table 2.4 shows IP major limitations, in four critical network functions: data processing, transmission, control, and storage. The first column identifies the function, the second column classifies the problems identified for that function, the third column explains the limitation in IP and the fourth column shows the NDN approach [15].

**Processing**. The IP router is not able to diagnose the data failures, it simply drops the content. However, NDN has a sophisticated mechanism to identify the reason for failure and taking the required actions. Also, NDN provides implicit content management such as re-locating contents by needs and increasing cache capacity/based on time-frequency, which IP cannot.

**Transmission**. The CDNs reduce the traffic-load in IP. However, the caching servers may not meet the requirements at a large Internet-scale. They have to be explicitly created according to demand. Also, the complexity of security requirements may lead to less efficient content distribution. Nevertheless, CDNs help mitigate the traffic-load of the Internet. The NDN is content-centric and overcomes CDNs complexity of transmissions with the caching role of each node. The NDN packets (Interest and Data) signed by consumer and producer, doesn't require middleware security protocols. However, the trust authorities may be needed, to build the integrity of data packets.

**Congestion Control**. Each NDN router records all requests and caches the contents, is able to reply to the next requests, even if one or more node fails. This brings less network congestion and it can control the network, which IP cannot.

[7] In this tool, the callee and caller are an effective content producer and consumer

Table 2.4: IP and NDN limitations between and comparison (adapted from [15]).

| Limitations | Classification of Problems | IP | NDN |
|---|---|---|---|
| Processing | Diagnosing failures | A host cannot identify the reasons of failure | Application layer identifies the failure |
| | Content and network handling | Internet services regulation leads to lack of content handling | Able to re-locate contents |
| Transmission | Efficient transmission of content oriented traffic | Current CDNs reduce the traffic-load but cannot meet with Internet scale. | Each NDN node caches content |
| | Security of transmission | Destination and source can be identified. | Content-centric |
| | Internet security | Several add-ons needed. | Certificate management required |
| Control | Congestion control | Only for end to end (some routers have limited capability for congestion control) | Each node is stateful, and records where the data is located. |
| Storage | Content aware network | Content only available in end-to-end points. | Each node caches the content |
| | Content security | Signed content without being re-requested again. | Application layer signs the content |
| | Content integrity | Content segments can be dropped and have to be retransmitted | Lost content segments can be completed by content store |
| | Content caching | CDNs have to request from publisher | If content is cached, no need to request from producer |

**Storage**. In NDN, routers cache the contents, signed by a producer, before issue as a data packet. Caching data packets also keeps content integrity, which eliminates segments drops during the transfer. In NDN, the content may be signed by the application layer, to resend content to the next-coming request.

The operational differences between IP and NDN can be classified as shown in Table 2.5. We have categorized six main differences (addressing, routing, forwarding, congestion control, security, and in-network caching).

**Addressing**. In IP, the packets identify the source and destination address. In NDN the packets refer to the data or interest name, not packet destination and source.

**Routing**. In IP (one-way traffic), the packets are not recorded in the routing table (except the address prefixes permanent table), which makes the forwarding process stateless. However, the permanent routing tables may cause an overhead (*e.g.* limited scalability) in large scale ISP topologies because of its one-way

Table 2.5: Operational differences between IP, CDN and CCN/NDN.

| **Operations** | **Base IP Model** | **CDN** | **CCN/NDN** |
|---|---|---|---|
| Addressing | IP | IP, Name | Named data |
| Routing (FIB) | IP prefix | IP, Name | Name prefix |
| | Single next-hop | Multi-path | Multiple interfaces |
| | Stateless FIB | Stateful FIB may be supported | Fully supported stateful FIB |
| Forwarding | Stateless packet forwarding | Unique State | Stateful forwarding at all nodes |
| | Packets not recorded in routing table | Packets may be recorded at replica nodes | Packets are recorded in every node |
| | Permanent routing table | Probe only replica nodes | Probe each faces at each node |
| Congestion control | Not supported | Supported only at replica nodes | Can be done in data plane by forwarding strategies, interest table, and content store in each nodes |
| Security | Only Channel | Channel and Content | Content-centric |
| In-network caching | Not supported | Not Supported | Supported by each node |

traffic. In NDN, the Pending Interest Table (PIT) stores the interest packets, while CS (Content Store) the data packets. In the meanwhile Forwarding Table (FIB) observes the content and records the network route. The FIB can handle the packet drops or identify legitimate interests by probing the router's interfaces.

**Forwarding**. In IP based networks, the forwarding interfaces are not recorded in the table. It is a simple design for forwarding without any observation, which makes the forwarding stateless. However, the cache providers (CDNs) can record their interfaces to decide: what packets to forward to which interfaces, and load-balancing data forwarding among the interfaces.

In NDN, each node can examine packets, that makes the forwarding stateful. For instance, the tables can register how many satisfied (data retrieved by interest)/unsatisfied (data not retrieved by interest) packets occurred, to measure performance. Also, based on a forwarding state, the adaptive forwarding strategies can be done by the network forwarder in NDN.

**Congestion control**. The one-way traffic may follow any route to reach the content producer in the IP network. This network behavior may lead to network congestion. The NDN overcomes (or mitigate) the network congestion because the router is only interested in content from the closest location. For instance, the congestion control can be done by PIT, CS, and FIB in NDN application layers.

**Security**. In IP, content security is achieved by securing the channel between hosts, and also the content itself. Therefore, the secured contents are only available for whoever is requested. The NDN security is content-oriented which makes the contents retrievable by next-coming requests.

**In-network caching**. The naive solution to increase data distribution is the use of CDN nodes answering the coming interests. In NDN, the content can be cached by any node, establishing the in-network caching.

## 2.8 Research Challenges

Similarly like the CCN network, the NDN offers data-centric authenticity, confidentiality, and integrity. The cryptography keys are used to bind names to a data packet. Because certificates are considered as a data packet, the NDN leveraged to address the research challenges of trust and key management.

NDN also offers the self-certifying name for per-packet data, which enables any node can verify published packet name that matches its content. In this case, the data name can be bind by the NDN application. Considering the names for application data, storage, certificate, routing and communication, the name-space management does stand as another research challenge as described by [3].

Lastly, the NDN architecture design poses content privacy challenges. Each node's cached contents may introduce cache size, cached names, and cached certificates in the content store. Considering each NDN node must have cached the content, the content privacy stands as another research challenge as pointed out by [16].

## 2.9 Summary

The NDN network proposes a transformation from today's Internet model to a new content-centric based architecture. It also promises maximized content distribution within application layers and protocols. Also, the application-driven based approach engages testing purposes and provides additional features to the original vision for the NDN research.

Since NDN is based on content (content-centric), the security is also content-driven. To provide integrity and origin authentication, a trust model is required. However, the NDN security has some operational challenges that make it as open research, such as the trust model used by application layer protocols.

Lastly, the NDN nodes (routers) use a Content Store to provide caching, so that may cause privacy issues about cache, content, name, and signature. This chapter briefly illustrated privacy challenges that can be threatening to cached contents by applying an adversary monitoring technique called a side-channel timing attack. The next Chapter 3 is focused on the side-channel timing attack and its possible countermeasures in NDN.

# 3

# SIDE-CHANNEL TIMING ATTACK AND COUNTERMEASURES

Privacy is a valuable asset that must be protected. It is usually regarded at two levels. Protecting specific user information, like age, sex, physical address, or identifiable individual information. Protecting information about user activities or sequences of activities, while using network applications. Extensive knowledge of this information makes individuals vulnerable to several other threats.

In this chapter, side-channel timing attacks and their effect on privacy are described, more specifically in the context of content-centric networks like NDN. Two real-time applications are used as an example: streaming over NDN (NDNtube) and Voice over NDN (VoNDN). Side-timing attacks and countermeasures can be tunned for each application. Then, a set of countermeasures, described in the literature, are presented, classified, and compared with each other. Most relevant related works are also described, either for NDN and for IP networks. The chapter ends with a discussion on the trade-offs of common countermeasures on content distribution.

## 3.1 Context

NDN is widely assumed to provide better privacy than point-to-point (IP) due to the former's lack of source and destination addresses. However, its content-centric design may maximize the privacy concerns compared to traditional point-point design [16].

In NDN, every node must cache transit data, to maximize the distribution of contents. When a cached content is re-requested by any consumer, the cache replies with content from the cache. Otherwise it lookups the content till its producer. Therefore, there is a slight time difference between cached and uncached contents. The adversary node or application uses this difference to know if a given content is cached by the router(s) in the producer path. This attack is called a side-channel timing attack, because the adversary is located side by side of the consumer, and they are connected to the same router. Through this attack, the adversary may affect the privacy of content, name, cache, and signature (certificate).

## 3.2 Side-Channel Timing Attack

The privacy-oriented side-channel timing attack can be based on information gathering from the computer implementation systems, rather than exposing software and algorithm weaknesses. The gained information, such as timing responses of the packet can be used to identify private information by an adversary. Also, the side-channel timing attack does not require any advanced configuration, because some information is publicly available and be retrieved by any network consumer [55].

In NDN, either or not the content is previously cached by CS, the consumer retrieves the packet in time which is called round trip time (RTT). It is defined as a time difference between the sent interest packet and received the data packet. Therefore, the RTT of cached content is shorter than the RTT of un-cached content. In an attack, the adversary takes advantage of the RTT differences to identify the cached and un-cached targets (contents) from the router.

Considering all nodes must cache the data, the privacy of cached contents can be potentially targeted by side-channel timing attacks in NDN. The works [16], [56], [57], [17], [21], [58], [53], and [18] discussed that the side-channel timing attack may affect the information privacy in NDN. Depending on the scope of the attack, mainly the adversary is able to identify the name, cache (*e.g.* size, content popularity), and the signature (*e.g.*certificate) through the side-channel timing attack.

### 3.2.1 Content Retrieval Time

In NDN, the RTT is used by routing strategies to find the optimal routes to send packets. These routes are chosen based on the shortest RTT between consumers and content providers. It is expectable that the RTT for cached contents is lower than the RTT for un-cached ones. Therefore, the adversary may use the RTT differences to identify the cached targets. In NDN, the RTT can be also used for other purposes, such as network performance evaluation.

In this work, the Content Retrieval Time (CRT) was defined to specify the RTT only for content retrieval. The CRT definition was used for cached and un-cached contents between the edge router and the consumer (adversary included). The CRT is the time between sending the interest and retrieving the content. The cached contents are replied from the content store of the intermediate routers between the consumer and the producer, and the un-cached contents are replied from the content producer.

In this work, the CRT calculation was adapted from the TCP/IP RTT estimation [59]. Figure 3.1 illustrates the CRT calculation from the edge router to the adversary node. In this example, the adversary sends “Interest 1” and retrieves “Data 1”, and then repeats again the same “Interest 1”. Then, the adversary analyzes the obtained $\mathrm{CRT}_1$ and $\mathrm{CRT}_2$ values to conclude whether “Data 1” was cached or un-cached from the edge router. The targeted “Data 1” is considered cached by the edge router if the adversary obtains $\mathrm{CRT}_1 = \mathrm{CRT}_2$, within very small error tolerance.

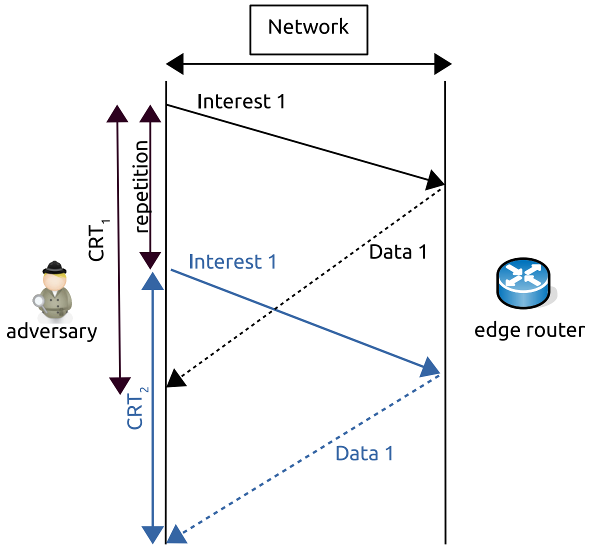


Figure 3.1: CRT measurements using one repetition of the same packet.

Whenever an interest is forwarded to the upstream node, the router starts a timer, which will be used to measure the CRT. When the data packet corresponding to the $\mathrm{n}^{\mathrm{th}}$ interest arrives at the router, this calculates the new CRT by the equation:

$$< \mathrm{CRT} >_{\mathrm{n}} = \gamma * < \mathrm{CRT} >_{\mathrm{n}-1} + (1-\gamma) * \mathrm{CRT}_{\mathrm{n}} \qquad (1)$$

where $\mathrm{n}$ is the number of received data packets and $0 < \gamma < 1$. If $\gamma$ is equal to $1 - \frac{1}{\mathrm{n}}$, then the real CRT is obtained. If $\gamma$ is close to 1, then the weighted average CRT is insensible to delay changes for a short time interval. If $\gamma$ is close to 0, then the weighted average CRT is very sensitive to new delay changes. These CRT calculations are presented for the same interest name sequence.

Note that, the adversary can use a different attack sequence of interests, depending on the attack design.

### 3.2.2 Attack Scope

In the NDN side-channel attack, the scope of the attack may vary by the intention of an adversary and targeted to a specific application (*e.g.* NDNtube and VoNDN). Based on the CRT values, information privacy can be threatened by the adversary as described by [16], [17], and [21]. In the side-channel timing attack, the adversary defines the targets by their content name (*e.g. /ndn-content*) or content segment (*e.g. /ndn-content/%00%12%34*). The adversary can target the following components: *i.* Name: The name of the content is defined by its producer and content itself. The adversary may also target the name to obtain the popularity of content. *ii.* Cache: When an adversary distinguished between cached and un-cached targets, this may also identify the size of the cache and top monitor the popular content(s) by their locations/regions. *iii.* Signature: In NDN, every data packet can be signed by its producer. This signature can be verified by certificates and these are also cached by routers. In this attack, the adversary may target the certificate to identify the user by measuring the CRT of the cached certificate.

**Name privacy**. In NDN, the name of the content can be threatened by the side-channel timing attack. The name of the content is not only visible, but it is also semantically related to the content packet and is used in routing protocols. In NDN, the requested data packet does not carry any information about who requested it. However, the targeted names may be used to obtain the popularity of the content from the cache of the routers.

Besides the content, the routing information, trust information, forwarding strategies are also named in NDN. Because the names are humanly-readable, any named segment or content can be targeted by an adversary. For instance, the large streamed content can be divided into segments and these cached by the routers. Also, the retrieved target may not be consumable by the adversary because the retrieved target is only a segment. Thus, the adversary only interested in the existence of the cached targets which are recently cached by the edge routers [53], [18], and [51].

**Cache privacy**. The targeted content names may also be used to identify or estimate the cache instruments such as size and policy. For instance, the LRU (least-recent-used) policy can be chosen for popular contents these can be cached by caching routers. Also, depending on the popularity of the content, the content producer may resize the cache to maximize the distribution for the particular popular content(s).

When an adversary identified the cached target by CRT, it may also identify the size of the cache by other identified targets. Through this attack, the adversary may know where the popular content has been recently cached and how long it stayed popular. The adversary can use this attack for content monitoring regional or country wise.

**Signature privacy**. The trust may be required on applications such as two-way communication application (*e.g.* VoNDN–voice-over NDN). In this application, the certificate authority may issue a public key certificate to the callee/caller for their integrity. Since NDN treats the certificate as like any other content, the certificates are also cached by the router. In this case, the certificates can be targeted and these may be used to locate the callee or caller in VoNDN. Ideally, the certificates can be produced by the certificate authority (CA) or self-signed. In large-scale applications, the trust may be maintained by a chain where the certificates are signed by Certification Authorities (CA). The certificates also publicly available which also helps the adversary to define the targets.

Let suppose, Alice and Bob exchange their certificates to establish a conversation. NDN certificates may be at least cached by one hour (freshness) if needed by Alice or Bob. In this attack, two adversaries can be located side by side to Alice and Bob. If the adversary successfully retrieved Bob's certificate from Alice router and vice-versa, the adversary can estimate the call time, when it establishes, and who established the call.

### 3.2.3 Attack Success Calculation

Through the collected CRT values, the adversary may distinguish the targets for the edge router or away routers. In order to understand better the attack, the attack success probability is discussed next. For this goal, let us consider the attack scenario illustrated in Figure 3.2. The consumer Alice (U) requests a content (C) from its Producer (P), and then C is cached by the edge router (R). The content C is signed by its producer P to keep its integrity. The Adversary (A) is probing the named C to learn if Alice (U) has requested it recently. If C is cached by R, then A succeeds on the attack. To distinguish the cached and un-cached targets, adversary A must probe the content C multiple times (at least two, to detect timing differences) increase the attack success rate.

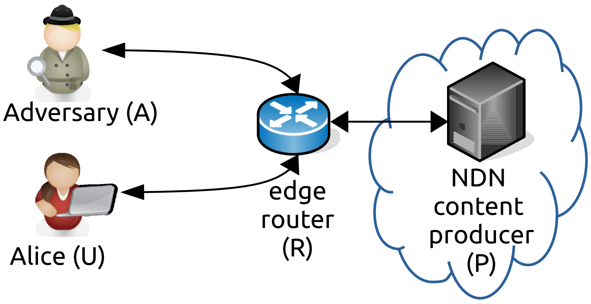


Figure 3.2: Side-channel timing attack on cached content.

The attack success calculation can depend on several aspects. For instance, the scope of attack and cache configurations may be different on different NDN applications (*e.g.* NDNtube and VoNDN) which can be an important factor for the attack success. In streaming, largely produced NDNtube contents are not structured as a single piece but split into content objects. In this case, the probability of attack may be increased because each segment must be cached by the edge router. In this case, the attack is designed by target prefix (*/ndntube/videos/video-1*), unlike the traditional segment-based attack design. When an attack is established, R replies to the request of A with a segment (*/ndntube/videos/video-1/%00%12%34*), which was previously requested by *U*. However, an adversary can increase the attack success rate by probing more than one content object from edge router R. The attack probability calculation is presented next.

Let us suppose that U sends the interest packets (*/ndntube/videos/video-1*) with a constant bit rate (CBR), P publishes the contents (*/ndntube/videos/video-1*) for U and these contents (including segments) are cached by R. The adversary A targeted a content segment (*/ndntube/videos/video-1/%00%12%34*) that has been recently requested by U and cached by R. Also, A repeats the targets to increase the attack success, for example, eight times.

Let us suppose a content object is sent in n segments and $\mathrm{P}_{\mathrm{failure}}$ is the attack failure probability for the adversary knowing if one segment is cached or un-cached. So, the probability of the adversary failing the attack with n available segments is $(\mathrm{P}_{\mathrm{failure}})^{\mathrm{n}}$. Consequently, the attack success probability of the adversary to know if one segment is cached or un-cached is expressed by:

$$\mathrm{P}_{\mathrm{success}} = 1 - (\mathrm{P}_{\mathrm{failure}})^{\mathrm{n}} \tag{2}$$

To illustrate the situation just described, a three nodes topology was created using the NDNtube application, as shown in Figure 3.2. According to the simulations that we ran in this attack scenario, using a 100 interest/s, the least recent used (LRU) policy, and a CS size of 100 packets, the obtained attack success probability was $\mathrm{P}_{\mathrm{success}} = 0.41$ for a single content object (*cf.* Figure 6.6 at Chapter 6). Consequently, the attack failure probability was $\mathrm{P}_{\mathrm{failure}} = 1–0.41 = 0.59$. So, if the content is split into eight segments and the adversary tries to retrieve at least one of the segments to have success on the attack, then the attack success probability is $\mathrm{P}_{\mathrm{success}} = 1–0.59^{8} \approx 0.85$, assuming the attack failure probability is the same for one segment and the single content object. Note that this attack success probability is calculated for an NDNtube application. For other attack designs and/or applications, the attack success probability can be of course distinct.

On the other hand, the sequence number may not be mandatory to establish an attack. Through our simulation experiences, the attack success probability can be higher if prefix matching is allowed from the application (*e.g.* NDNtube or VoNDN). For instance, instead of requesting a certain sequence number (*e.g.* */ndntube/videos/video-1/%00%12%34*), the adversary can request a prefix (*e.g.* */ndntube/videos/video-1*) and router replies the request with its segment (*e.g.* */ndntube/videos/video-1/%00%12%34*).

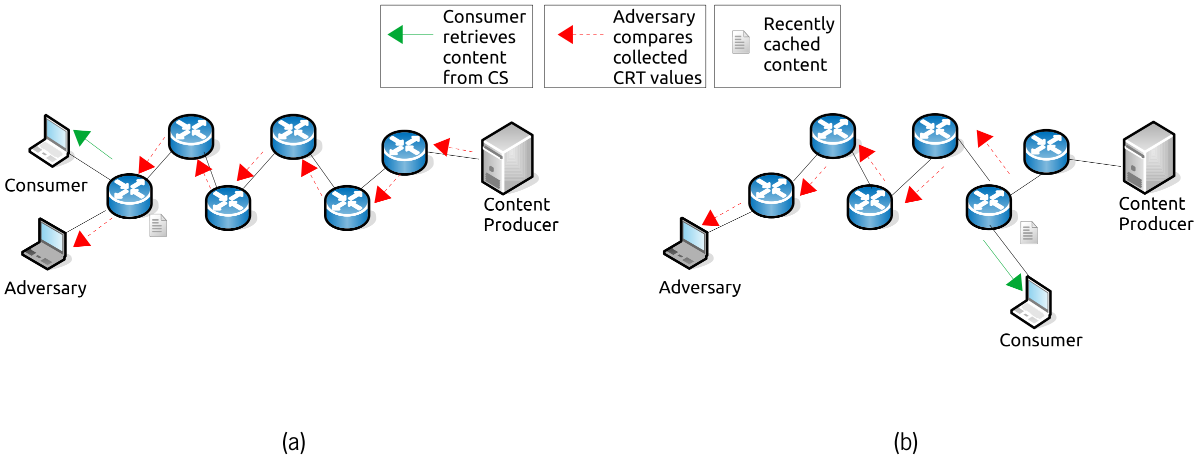


Figure 3.3: Attack design and scopes: (a) Identifying closest hop cached contents. (b) Identifying distance hops cached contents.

The CRT side-channel timing attack can be designed based on two adversary models: *i.* one able to identify the target at the edge router (Figure 3.3a), and *ii.* another able to identify the cached targets from away routers (Figure 3.3b). When an adversary collected all possible CRT values, it can take three different assumptions about where the target has been cached.

First, the maximum CRT value shows that the target is not cached by any NDN routers. Second, the minimum CRT value indicates the target has been recently cached by the edge router. Third, if the CRT is between a minimum and a maximum value, the adversary concludes that the content was cached by an away router.

Additionally, more than one adversary can be used to determine the established call between Bob and Alice when adversaries get the shortest CRT for their certificates, as illustrated in Figure 3.3a. On the other hand, a single adversary may still determine targets from away location, as illustrated in Figure 3.3b.

## 3.3 Side-Channel Timing Attack on NDN Applications

This section presents the possible attack models applied in NDN applications. In this work, the attack models were focused on two NDN applications: *i.* streaming application, which the application has a single producer and many consumers; and *ii.* voice-over NDN, which present two-ways communication between two peers.

### 3.3.1 Streaming over NDN

A streaming media user can be listening to media or watching a video in real-time or pre-recorded over the Internet (IP) by current streaming applications. In streaming, continuous content is delivered by the producer to the consumer. The video may be delivered to be saved in the cache for later on-demand playback by CDN caching servers. In this way, a streaming producer may handle failures and diversity, such as traffic congestion, multiple versions of an encoder, different device video resolution, *etc.*

**Architecture of NDN streaming applications**. The video distribution applications benefit from the NDN architecture. The live streaming tools (NDNlive, NDNvideo, and NDN-RTC) and pre-recorded and live stream tools (NDNtube) were proposed by [6], [60], and [61], and [43]. Instead of relying on centralized servers as it is constructed with current streaming applications, the NDN design may make the servers robust by naming the streaming packets, which can be independently retrieved from the network layer. Through NDN, the applications fetch the streamed content by names, and the content can be delivered either by the producer or by any router's CS. That also removes the third-party application requirements of managing and locating streamed contents as designed in NDNtube [61] and video-conferencing applications [43].

The audio conference tool (ACT) architecture over NDN, proposed by [62], takes the advantages of named data to locate the conference, the speakers, and to fetch packets from speakers. The tool also announces the conference by using a signaling protocol, which is called the Session Description Protocol (SDP) [63]. The name is constructed as */ndn/broadcast/conference/session/speaker-list*, for example.

When the voice packet is generated by the speaker, the ACT server caches the data in SDP format, which describes the name prefix that can be used for media type, voice data, and public key locator.

$$\underbrace{/\mathrm{ndn}/\mathrm{pt}/\mathrm{uminho}}_{\texttt{routable prefix}}\ \underbrace{/\mathrm{NDN\_x}}_{\texttt{application}}\ \underbrace{/\mathrm{stream\_1}}_{\texttt{stream ID}}\underbrace{/\mathrm{video}}_{\texttt{media}}\underbrace{/\mathrm{content}}_{\texttt{content}}\underbrace{/\mathrm{frame\_num}}_{\texttt{frame}}\ \underbrace{/\%00}_{\texttt{segment}}$$

Figure 3.4: Video streaming and audio Packet Format.

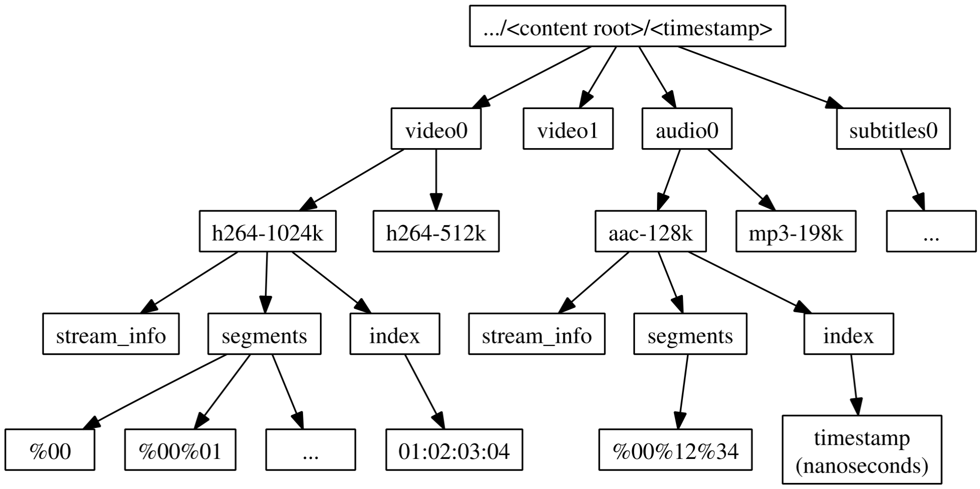


Figure 3.5: NDN streaming applications namespace (adapted from [6]).

Figure 3.4 and Figure 3.5 illustrate a generic name hierarchy on NDN streaming applications. The large video content is split by video segments with its frame number. The NDN video and audio name components are presented by the following:

**routable prefix**: This is used to identify the interest and to forward it in the NDN network.

**application**: The NDN_x is used to identify the NDN application name such as live streaming, video, RTC, and Tube.

**stream ID**: This is an identifier used to distinguish one stream among the others.

**media**: This is used to identify the content type for /video and /audio.

**content**: The audio and video frames are structured in the content and it may be also used to identify the streaming information, such as codec H.264 and metadata.

**frame**: It identifies each audio and video frame by a number.

**segment**: It identifies each data frame segment.

The content name may also include a content verification parameter and is identified by the MetaData section of a data packet, as described by [44]. Each frame is signed by the producer and signature carried within namespace `<stream>/key` in MetaData. The MetaData parameter may also include a freshness period (*e.g.* ≈1000ms) of the data packet, which also defines the cache time for the data packet by its producer.

**Possible attack scopes on NDN streaming applications**. The pre-recorded and live contents can be published by NDNtube, NDNvideo, and NDNlive applications. To answer live and future requests when the producer becomes off-line, the video and its segments can be cached by routers. Through this approach, the load of the producer can be reduced and the video distribution maximized [6], [61].

The audio/video segments are cached by the CS and each segment can be targeted by a side-channel timing attack. The video segments can be targeted by an adversary to obtain the location of the targets. The attack can be designed for a single segment by an adversary to identify the popularity of video. Also, the attack can be configured to monitoring the cache to obtain the video types or contents by the region.

### 3.3.2 Voice over NDN

Voice over IP (VoIP) is a transmission method of voice/video communication over IP. VoIP requires intermediary exchange protocols, such as Real-Time Transport Protocol (RTP) or Secured RTP (SRTP), and a signaling protocol, *e.g.* Session Initiation Protocol (SIP) to establish a call, as described in [64].

**Voice over CCN (VoCCN)**. In CCN networks, the data flows directly from the producer to the consumer. Therefore, the media and signaling paths can be defined between the producer and the consumer. Based on this idea, the VoCCN design has been proposed by [7]. In VoCCN, the signaling and media paths can be combined and the voice packets can directly flow between callee and caller without requiring any translation middleware because packets can flow directly between callee and caller.

**Architecture of Voice over NDN (VoNDN)**. Similarly to VoCCN, this work introduces voice-over NDN (VoNDN) as a use case for testing purposes.

As Figure 3.6 illustrates, the NDN name can be used to establish signaling and media paths for voice/video calls. The SIP may be used to create a signal path from Alice to Bob. A SIP invitation message carries a randomly generated symmetric key *k*. The caller (Alice) can encrypt the key block *(k)* using callee's (Bob) public key (B_pub). When creating a signaling interest packet the caller would include both the encrypted block (B_pub(*(k)*)) and the authenticated SIP message (*(k)*(SIP_INVITE)). The callee, on receiving the interest, could decrypt the key block with its private key, recover *(k)*, and use it to verify and decrypt the SIP_INVITE. The caller would then use key *(k)* to encrypt its SIP_RESPONSE message[1].

When the signaling path is securely established, the SIP packets are replaced by RTP media packets. As seen in Figure 3.6, the SIP exchange section is replaced by the call identification (call-id), together with other required information and a sequence number (seq-no) used to control different media fragments.

**Possible attack scopes on VoNDN**. Theoretically, the secured VoNDN conversation may face side-channel timing attacks. The aforementioned attributes of encrypted traffic (trusted conversation) of the side-channel information may be used to leak insights from the communication users. Since the callee and caller are presented as producer and consumer, the cached contents may be used to identify the callee

[1] In VoNDN, the *callee* and the *caller* play the role of producer and consumer at the same time.

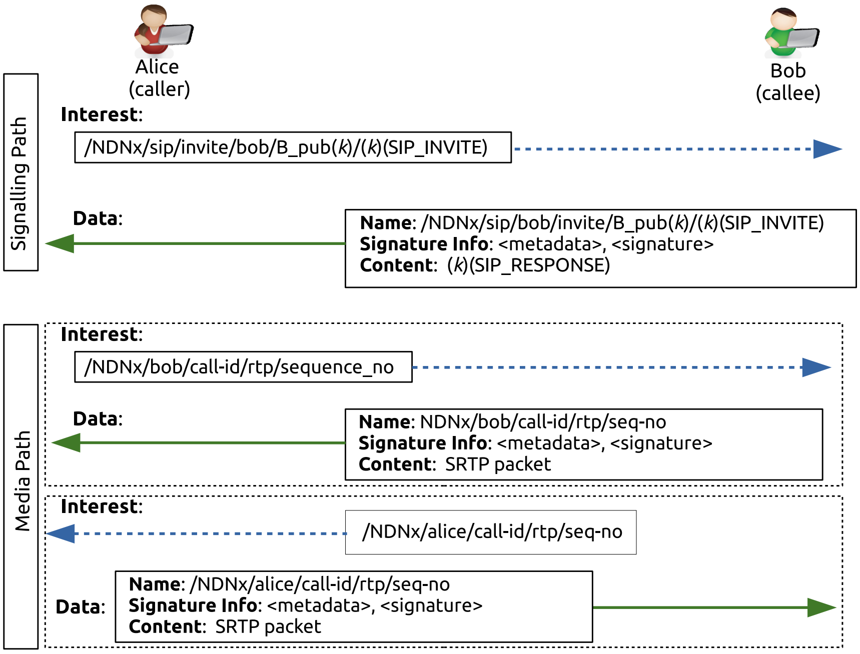

Alice
(caller)
Bob
(callee)
Signalling Path
Interest:
/NDNx/sip/invite/bob/B_pub(k)/(k)(SIP_INVITE)
Data:
Name: /NDNx/sip/bob/invite/B_pub(k)/(k)(SIP_INVITE)
Signature Info: <metadata>, <signature>
Content: (k)(SIP_RESPONSE)
Media Path
Interest:
/NDNx/bob/call-id/rtp/sequence_no
Data:
Name: NDNx/bob/call-id/rtp/seq-no
Signature Info: <metadata>, <signature>
Content: SRTP packet
Interest:
/NDNx/alice/call-id/rtp/seq-no
Data:
Name: /NDNx/alice/call-id/rtp/seq-no
Signature Info: <metadata>, <signature>
Content: SRTP packet

Figure 3.6: VoNDN combined paths (adapted from [7]).

or the caller (conversation pairs), the location, and the time of the established conversation. For instance, Zhang *et al.* [8] shows how to reveal the voice call history of the user by side-channel timing attack in IP. The attack method also can reveal the call history of a group call in VoIP. The timing attack is aimed against the victim's SIP proxy server. The other work Lauinger *et al.* [65], [56] studied the side-channels on Voice-over CCNs. It is shown that an adversary can replicate the VoCCN packets to learn the size of the cached voice packets. Because the voice is encoded using by variable-bit-rate encoding scheme, each voice packets can be shaped by its phrases. This may lead to learning the previously spoken voice packets even the conversation is encrypted between VoIP pairs as described by [66], [67].

In VoNDN, the voice packet public-key integrity is established by the certificate and may be managed by a SIP authentication domain, such as an inter-domain authentication protocol (*e.g.* IP authentication protocol RFC4474 [68]). To expedite the next request(s), the certificate is cached for a certain period by a SIP proxy.

As Figure 3.7 illustrates, the certificate from the caller's domain is cached by callee's proxy. The adversary takes advantage of the SIP processing time to obtain a certificate that has been cached or not. Through the time responses of certificates, the adversary may obtain the VoNDN call history of a SIP domain.

In this work, the Content Retrieval Time (CRT) is defined as the period between sending the interest and retrieving the respective content, which can be cached or un-cached. As certificates are treated as contents, the CRT definition can be also applied for the certificates.

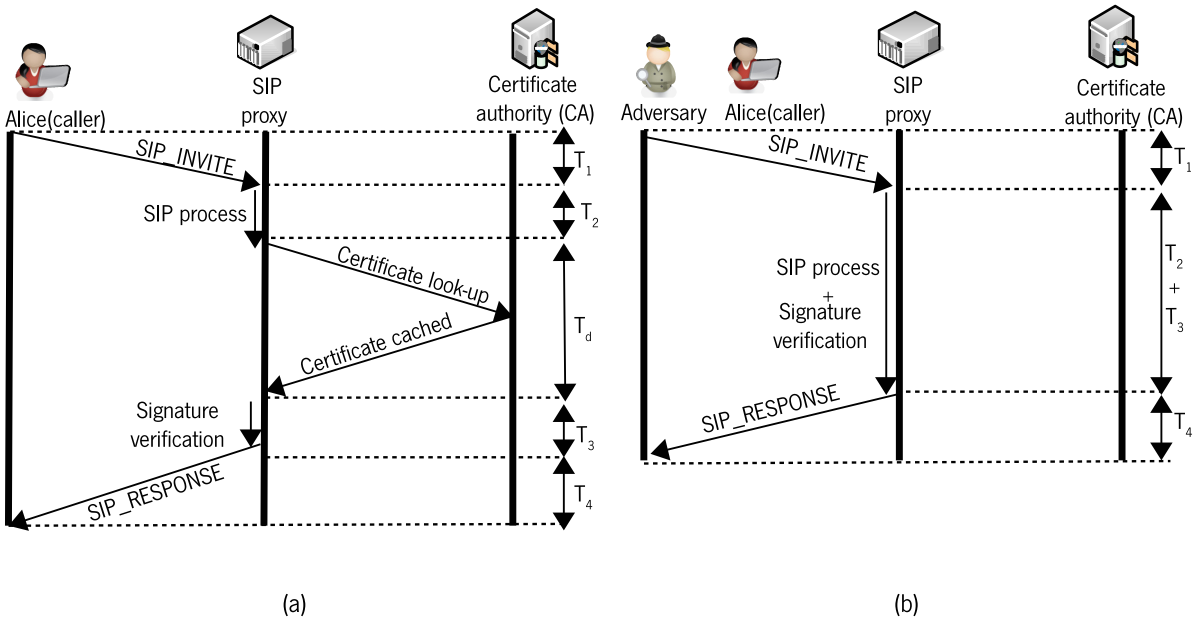


Figure 3.7: Side-channel timing attack on VoNDN trust scheme: (a) First time for certificate lookup, (b) Future request the certificate from CS. (adapted from [8]).

Figure 3.7a illustrates that the caller has to get the certificate from the certification authority since it is not cached by CS. This certificate request process time is calculated by equation:

$$\mathrm{CRT}_{\mathrm{uncached}} = \mathrm{T}_1 + \mathrm{T}_2 + \mathrm{T}_{\mathrm{d}} + \mathrm{T}_3 + \mathrm{T}_4 \tag{3}$$

where:

$\mathrm{T}_1$: the content retrieval time for sending the SIP_INVITE message.

$\mathrm{T}_2$: the processing time for the SIP message.

$\mathrm{T}_3$: the signature verification time for Alice's identity.

$\mathrm{T}_4$: the CRT time response from Alice SIP proxy (*e.g.* content store).

$\mathrm{T}_{\mathrm{d}}$: the CRT time for the intended lookup certificate.

As illustrated in Figure 3.7b, if Alice's certificate has been cached by CS, then its request process time is calculated by the equation:

$$\mathrm{CRT}_{\mathrm{cached}} = \mathrm{T}_1 + \mathrm{T}_2 + \mathrm{T}_3 + \mathrm{T}_4 \tag{4}$$

In this example, the adversary targeted Alice's certificate by distinguishing the cached and un-cached certificate CRT responses.

Note that, only the caller's (Alice) side is illustrated in Figure 3.7. On the other side, similarly, the callee (Bob) receives the invitation from the caller that needs to be verified and fetch with the caller's (Alice) certificate from the SIP server. Thus callee can validate the signature of the caller to establish the call.

**Determine close and away targets**. In naive condition, the certificate can be replied from the corresponding CS to the callee and caller in VoNDN. The adversary may use cached certificate CRT value to determine the consumer location or established call time. Note that, if the certificated packet is only issued for a certain consumer, the adversary cannot determine the content because of a lack of the private key. Still, the adversary can knowledge the existence of the location of the cached content because of knowing the public key.

On the other hand, the adversary can determine the distance of a certificate from its cache using the CRT information. Let us suppose that the side-channel timing attack CRT measurement for a certificate is $\text{CRT}_2$ (retrieved content from CA), $\text{CRT}_1$ is the CRT from the edge NDN router, $\text{CRT}_\text{e}$ is the expected CRT of the intended content lookup, and $\varepsilon$ is a very small time difference. According to Chaabane *et al.* [53] and Mohaisen *et al.* [18], after collecting the CRT samples, the adversary concludes that:

- if $|\mathrm{CRT}_\text{e} - \mathrm{CRT}_1| < \varepsilon$, the target certificate has been cached by the edge router.
- if $|\mathrm{CRT}_\text{e} - \mathrm{CRT}_2| < \varepsilon$, the target certificate is not cached by any router, except the certificate authority.
- if $\mathrm{CRT}_\text{e} > \mathrm{CRT}_1$ and $\mathrm{CRT}_\text{e} < \mathrm{CRT}_2$, the target certificate has been cached by away routers. Note that, the adversary can still predict the certificate location (number of hops) by relying on $\text{CRT}_1$ and $\text{CRT}_2$ values.

## 3.4 Countermeasures

In NDN applications (NDNtube and VoNDN), the side-channel timing attack can be mitigated by statically (always-on) pre-configured countermeasures on the caching routers. These can be based on the manipulate of the CRT values of the NDN router. Through this, the adversary may not able to distinguish between cached and un-cached CRT values which can be used to obtain the cached contents or the certificates. In this work, the countermeasure methods were classified by *i.* cache available and *ii.* cache disabled approaches.

Figure 3.8 illustrates the main concept of the countermeasure approaches for NDNtube and VoNDN applications. In NDNtube, the adversary attempts to retrieve a segment that belongs to a video. On the other hand, the adversary focuses a certain named certificate in trusted-NDN applications (*e.g.* VoNDN).

Figure 3.8a shows Alice requested content from the streaming producer and it replied the content segments, were cached by the router and CRT calculated as $\Delta_1 + \mathrm{t}_1$. If Alice re-request a segment, the

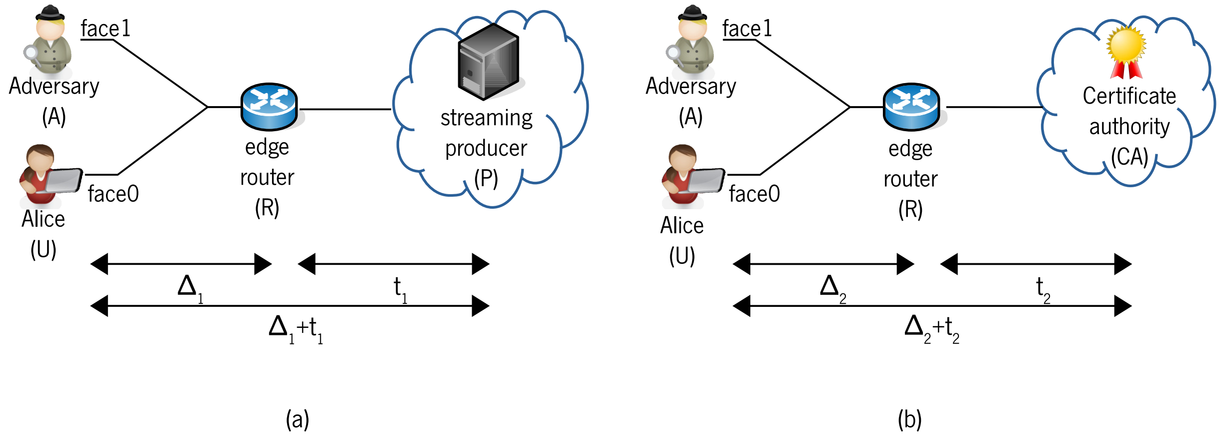


Figure 3.8: Statically configured countermeasures. (a) NDNtube countermeasure configuration, (b) VoNDN countermeasure configuration.

cache replies to this request instead of sending it till the producer, and the CRT is calculated as $\Delta_1$. In this scenario, the adversary pursues the CRT of $\Delta_1$ to obtain the target that has been cached recently.

Figure 3.8b shows a similar attack scenario on the trusted VoNDN application. The adversary follows the same procedure to succeed in the attack for the targeted certificates.

Since the $\Delta_1$ and $\Delta_2$ can be used to illustrate for all cached contents and the adversary pursues it, the countermeasure methods can be based on increasing the CRT value of $\Delta_1$ and $\Delta_2$ with some additional value from the configuration, as discussed next. Based on the countermeasure configurations, the attack can be mitigated on the adversary face (face1). However, these countermeasures are based on the static configuration which also affects the legitimate node requests (face0). Therefore, the pre-configured countermeasures are not able to distinguish between adversary and legitimate nodes.

### 3.4.1 Cache Available Methods

The cache available countermeasure methods are used to increase the value of $\Delta$ to all faces (face0 and face1) and these can be classified into three groups: *i.* delay content, *ii.* random caching, and *iii.* group signatures.

**Delay content**. Data delivery in the NDN is affected by a certain delay imposed by the routers. This delay can be a solution to prevent cached content attacks. Let us consider $\Delta$ the default delay value which presents the CRT of cached targets. In a side-channel timing attack, the adversary tries to figure out the $\Delta$ value. The adversarial CRT calculation can be challenged if a delay of $\tau$ was chosen based on a random function by the router. In this case, expected CRT is increased for the adversary, which concludes that the target has not to be cached by the edge router. However, an additional delay to $\Delta$ may reduce the content distribution for the cached contents.

On the other hand, because of the attack repetitions, the adversary may find a delay of $\tau$ by analyzing the CRT samples from the router. The value of $\tau$ can be changed by proposed algorithms. For instance, instead of using a constant $\tau$, Schinzel and Sebastian [69] proposed a $\tau$ value based on cryptographic (unpredictable) function. Through, the unpredictable delay function of the $\tau$, the adversary may not able to obtain the cached targets in the router.

A similar delay countermeasure method was proposed by work Acs *et al.* [17] to preserve privacy in NDN. The delay was applied to all faces to mitigate the timing attack in NDN. This delay can be classified by three configurations: fixed, randomized, and unpredictable. To improve distribution efficiency, the delay is configured only for the first requests. Therefore, the first adversary's requests miss the cache and the attack may not succeed for the cached target as proposed by [17] and [53].

**Randomly caching**. To reduce the cache redundancy, the capacity of the cache can be defined using probabilistic order as presented by [70]. On the other hand, the probabilistic cache can be used to mitigate the side-channel timing attack. The router can be configured for randomly caching, one named as may be cached another may not be cached depending on the probability configuration. The contents can be also cached by random anonymity set (*k*), to mitigate the side-channel attack in NDN. Acs *et al.* [17] and [53] proposed a random caching, that selects the contents by depending on the random number (*k*) to mitigate the attacks on the edge router.

Thus, the index of the first cache hit in the output sequence is expected to be random and ideally should not leak information about the router's cache. However, the adversary may also learn the anonymity set value after various cache miss attempts. Therefore, the random value can not be fixed, it should rely on various *k*-anonymity set (*e.g.* probability rate).

**Group signatures**. In a side-channel timing attack, the adversary uses the CRT estimation to know where the certificate is cached. Cham *et al.* [71] proposed a *group signature* to make public signatures (*e.g.* public key certificate). The signature and the certificate can be associated with a group of users, but not to a specific member of that group. The receiver knows that the signature is valid for any group member. For instance, a conference video/call may use a group signature for privacy protection because the adversary cannot know which member of the group is doing the call.

The other work Boneh *et al.* [72] proposed a short group signature scheme. In this approach, the main goal is to provide the security level of RSA signatures while reducing the length of the signature to accelerate the verification in the group.

In both methods, the certificate can be cached only with a group member to achieve perfect privacy for the certificate.

### 3.4.2 Cache Disabled Methods

The cache disabled countermeasure approaches offer a "*perfect privacy*" by fully supported anonymity tools. However, the disabled caching approaches can be completely against the NDN paradigm, considering the content distribution must have to be maintained with in-network caching on NDN. In disabled cache approaches, the CRT is obtained as $\Delta + t$ (Figure 3.8), which is considered as the maximum delay for Alice (face0) and Adversary (face1) faces.

**Turned-off caching**. In NDN, the CS can be configured for not caching. If there is no content held in the cache, the side-channel timing attack cannot be done. However, the cache is important for the NDN, as it is required for content distribution. So, directly giving up on the caching is not a good option in NDN, as discussed in [17].

**Anonymous Named Data Networking Application**. The Onion Routing (Tor) was employed layers of concentric encryption and intermediate nodes responsible for peeling off layers as packets travel through the overlay which is commonly referred to as onion routing as proposed by [73]. DiBenetetto *et al.* [21] developed the Anonymous Named Data Networking Application (ANDaNA) a tool to mitigate timing attacks in NDN. ANDaNA is another practice of Tor, built on top of NDN, that provides privacy and anonymity to the consumers. With this tool, the requested names are encrypted and then verified by the nodes and delivered to the user as data. In particular, ANDaNA mitigates timing attacks from linking the retrieved contents in CS. ANDaNA relies on multiple paired-centric layers of encryption and routes content from the consumer via a chain of routers. First, the router decrypts received content then it forwards the content to the next router.

**PrivICN**. The PrivICN is a tool based on name encryption similar to ANDaNA. The tool encrypts the name components except for the longest prefix of the content as presented by [19]. Therefore, the cache is partially available only for the longest prefixes. However, the adversary still can succeed in the attack considering the longest name prefix target to locate target locations.

**Bloom filtering**. The name privacy can be maintained by bloom filters as presented by [53]. In this approach, the consumer can compute hierarchical bloom filter as HB = $(B_1, B_2, ..., B_n)$, where $B_n$ is the bloom filter of name component up to n-th component. For example, a consumer can compute a filter $B_1$ of */ndn*, $B_2$ of */ndn/pt*, and $B_3$ of */ndn/pt/minho* for the content of */ndn/pt/minho*. Thus, a router can check the filter $B_n$ from the cache, if it cached it replies to the consumer. If not, the router checks $B_n$ in PIT. If $B_n$ existed in PIT, the bloom filter of the corresponding PIT is updated (add one) and the interest dropped since a request has already been forwarded. Otherwise, it follows the usual NDN paradigm. With this approach, the name in the interest request is obfuscated resulting in transforming it into a random string of bits. However, bloom filters can introduce false positives in name matching.

Kondo *et al.* [74], also presented a similar filter-based approach to preserve name privacy in NDN. This work distinguished the legitimate requests from others based on the filtering. Through the bloom filters, the name of content becomes unreadable because the human-readable name was transformed into a random-looking string of bits.

## 3.5 Related Works

In this section, the related works were presented based on the side-channel timing attack and countermeasures on NDN and point-to-point protocols. Also, the scope of the attack was analyzed by the comparison of NDN and IP.

### 3.5.1 NDN Related Works

The privacy issues were discussed by several NDN research works. The following works were surveyed these mainly focused on NDN security and privacy.

**Attack related works**. The work [75] presented an attack-type that is for Geo-locating the consumers in the NDN-testbed. The consumer may have the information about the hop count that is used to obtain the hops between the routers. The adversary may use this information to obtain consumers' cached contents by NDN-testbed hops. This attack is similar to a side-channel timing attack because the cached contents hop counts can be slightly noticeable for non-cached contents. To mitigate the attack, it was prosed that the hop count information may be turned-off for the users.

The works [56], [16], [18], [57], [65], [21], and [17] discussed the side-channel timing attack and its countermeasures in ICN and NDN. The traditional attack models were presented to design the adversary application. Also, different countermeasures were proposed to mitigate the attack which will be discussed next.

**Countermeasure related works**. The work [17] widely studied the cache privacy and the adversary threats to the consumer and producer privacy on the NDN paradigm. Also, countermeasure methods were proposed to mitigate the attack based on the other work [20] -*k* anonymity based delay algorithms (no cache, delay, and random cache). These countermeasure methods applied to privacy-sensitive contents can be indicated by its producer and consumer.

The delay may be used to mitigate the side-channel attack on privacy-sensitive indicated contents. However, an additional delay may imply a trade-off between privacy and latency because it is also applied to legitimate requests. For instance, a higher $\tau$ value can disable the cache on routers as discussed by [17] and [53]. Also, user-driven countermeasures may not be usable in the real world. For instance, what is private for a user may not be private by other users.

The work [18] presented an extensive study for timing attacks on ICN privacy. The side-channel timing attack and its findings were presented with possible countermeasures methods to mitigate the attack in ICN networks. The trade-offs of countermeasures were evaluated by primary results, especially on additional delay approach algorithms. Also, they proposed a user-driven countermeasure method called "*Vanilla*". For privacy-sensitive contents, an edge router caches the content from the producer and keeps the retrieval times of the first interest and delay the next coming requests. However, the per-client solution will not be feasible, because of the large number of consumers and content distribution efficiency.

The NDN promising maximum in-network caching feature to achieve lower latency for requested contents. However, the work [76] stated that the cache may not be necessary to be configured for maximum size. For instance, the cache can have the same performance on different distributions. Through not caching each content, privacy can be preserved for a timing attack. This method can reduce the attack performance but considering at least one segment must be cached by a router, this can be still targeted by an adversary.

The work [21] presented an anonymity tool called ANDaNA that is built on top of NDN. The tool provides maximized consumer anonymity through unreadable (encrypted) name-spaces. However, when the name-spaces became unreadable except for whoever asks for it, the usage of CS becomes useless because no consumer can retrieve contents from CS. Similarly like ANDaNA, the work [19] also presented a system tool called PrivICN that relies on an encryption scheme on contents. In the other work Kondo *et al.*[74] and [53] proposed a name filtering against information leakages in NDN. Also, the work [53] discussed that bloom filters introduce false positives and periodically require resetting and reducing the performance of the cache.

Effectively, the un-readable name and filtering techniques may prove "perfect privacy" but also comes with the disabled cache. Since NDN promising the contents must be cached to achieve low latency, the name filtering and encryption approaches may not be the most feasible for the NDN paradigm.

The group signatures can be used to preserve the signature (certificate) privacy as proposed by Cham *et al.* [71] and Boneh *et al.* [72]. Consequently, the privacy of the certificate or content can be maintained in the trusted group. However, considering collaborating with the group members, such a group trust scheme relies on limited configurations for verification and identification.

The work [57] studied possible privacy risks and their countermeasures cost of the performance in NDN. The work also studied possible naïve countermeasure methods such as selective tunneling, selective caching, and attack detection to overcome countermeasure performance issues.

**Surveys**. The surveys [53], [51], [50], [49], [77], and [78] studied the security and content privacy threats/countermeasure methods in other future Internet architectures and Information-Centric networks. Consequently, the related surveys indicated the side-channel timing attack can be a privacy threat to future Internet architectures because these promising the in-network caching.

The countermeasure methods also addressed by the survey works. In brief, these were classified by no-cache, delay with different (-*k*), randomly caching, and the unreadable content names.

### 3.5.2 IP Related Works

The consumer-driven approach is implemented using the work [20] -*k* anonymity based delay algorithms (no cache, delay, and random cache) to web browser histories. The k-anonymity delay-based algorithms can also be applied in NDN routers. Through these router manipulations, the adversary may not identify the cached contents in NDN. However, the countermeasures can be affected by the CS performance, because the response of contents has an additional delay. This approach is a trade-off between privacy and latency. To the best of our knowledge, each of the contents has to be considered as private rather than distinguish the contents for private and non-private.

The work [69] presented the side-channel timing attack on web applications and proposed several countermeasures to mitigate it. The countermeasure methods were based on response time and this was classified by fixed, random, and unpredictable delays on web applications.

The work [8] proposed a side-channel timing attack, to expose calling history in VoIP services. The SIP providers may rely on the user certificate model, to keep the integrity of communication between end-users. For instance, the caller's certificate can be cached by callee's proxy server to accelerate the next requests. The SIP response can be different from each other because of cached certificates. The adversary uses the timing attack by probing content requests to identify caller and callee call history. Similarly like other works, they also proposed additional delays for SIP responses for VoIP service providers. The other [79] study, created a timing attack model for tracing the VoIP encrypted calls over the Internet. The adversary tracks watermarked packets between VoIP peers.

The work [80] presented an attack-type on electronic gadgets these are similar to a side-channel timing attack. The adversary deploys the attack on cache memory of a particular TV box and sport tracking kits. Also, the devices are using encryption to protect their contents. However, the products may offer the transmission characteristics from various data rate encoding which this information may be identical for particular media content. Therefore, the adversary may conclude that what movie has been played by caching a simple encoded movie title from the TV box.

The work [81] studied the timing attacks and classified them into two types; direct timing and cross-site timing attack on the point-to-point protocol. The adversary measures the HTTP request responses to identify recently cached websites.

## 3.6 Discussion

In this Chapter, possible cache privacy-related side-channel timing attacks and its countermeasures are presented. It's shown that the time difference between cached and un-cached content can be used to determine sensitive information regarding content name, cache size, and signature.

To show attack findings, possible attack scenarios are studied on NDN applications (streaming and VoNDN). In steaming NDN applications, an adversary can determine the popularity of the streamed content by targeting the streamed video segments (*e.g.* NDNtube). This attack also is used to monitoring the cache to determine the streamed video types or contents by the region. On the other hand, an attack can be a threat considering user privacy by determining the cached certificate location in trusted NDN applications (*e.g.* VoNDN). An adversary obtains sensible information such as the location and the time of the conversation between callee and caller in trusted-VoNDN.

The attack can be mitigated by previously studied countermeasure methods on the NDN applications. In this Chapter, the countermeasures are categorized by cache available and disable methods. These methods are based on the time manipulation of the retrieval time (CRT) between user and router. Through these methods, an adversary may not able to distinguish between cached and un-cached targets.

The related works presented by point-to-point and NDN works in sense of their attack models and countermeasures. It is shown that the current countermeasures are considered as a trade-off between privacy and content distribution efficiency.

# 4

# ATTACK AND PRIVACY MODEL DEVELOPMENT

The side-channel timing attack can be used to identify previously cached contents effectively. However, the traditional attack designs may be considered as old-fashioned and inefficient compared to today's attack designs. Thus, in this chapter, the brute-force (burst-like) method is adapted (re-designed) to the side-channel timing attack to improve the success of the attack.

To mitigate side-channel timing attacks (traditional and brute-force), the countermeasure methods were discussed previously. However, these statically configured method(s) may not be efficient approaches considering the NDN content distribution. In this chapter, to apply the countermeasures only under the attack the detection methods are discussed and an approach called detection and defense (DaD) is presented.

## 4.1 Context

This chapter introduces the main contribution of the work, by presenting: *i.* design an attack model that is based on brute force to increase the attack success compared to traditional designs, and *ii.* an efficient countermeasure model that based on detection to mitigating the brute-force based side-channel timing attack.

The traditional attack models may not be the most efficient ones when multiple targets are considered. Therefore, in this work, a method called brute-force was adapted to a side-channel timing attack to engage in multiple targets. Also, additional attack configurations and scope of attacks were discussed to increase the success of the attack.

The side-channel timing attack can be mitigated by previously configured countermeasures such as unpredictable delay, random, and no-cache configurations. However, these configurations may affect legitimate requests from the cache performance. To protect legitimate requests and improve the content distribution performance a novel privacy model called detection and defense (DaD) is introduced in this chapter. The model intents are that the proposed countermeasures can be only applied faced to the existence of an attack. To achieve this design, attack detection methods also are proposed.

Based on the above discussion, the main contributions of this work are summarized in the following items:

- A novel attack model designed based on brute-force to increase attack success rate compared to traditional attack.
- Discuss possible attack scope for NDNtube and VoNDN applications.
- Propose detection methods for adversarial faces in a side-channel timing attack.
- Propose a privacy model that is based on the detection of the attack and its severity, then apply a multi-level countermeasure method.

## 4.2 Brute-force attack development

In traditional side-channel timing attack design, an adversary defines the targets for objects of particular content (segments). However, this type of attack uses to be inefficient in terms of success, when it is directed simultaneously to multiple named streamed content or certificate.

In this work, an attack design called brute-force was adapted to the side-channel timing attack. Using a brute-force attack, the adversary can target multiple contents and attack them in a short period. This attack can be also considered as a burst attack that is using repeated short bursts of targets at random intervals [82].

In the brute-force attack, the adversary tries all possible combinations of password dictionaries until getting one that matches [83], [84]. To mitigate this attack, the web providers limit the requests for a short period (e.g. several failed passwords attempts) and enhancing the complexity of the password dictionary (e.g. requiring special characters).

In this work, the brute-force was configured for the content names measuring the response times from the edge router. In this attack, an adversary defines the targets by content segments trying to retrieve them in a short time. These targets can be streamed content segments or public-key certificates which can be defined by the requirement of the attack. Additionally, this brute-force attack was designed to retrieve streamed content or certificate by randomly and at the same time in order to increase the attack success.

Next, this work's brute-force design algorithm and its random probing function are presented.

### 4.2.1 Attack Procedure

Figure 4.1 illustrates the brute-force attack process to success the attack for the multiple targets $(\mathrm{T_n})$. These can be the predefined name of content or certificate to start the attack procedure for data packets.

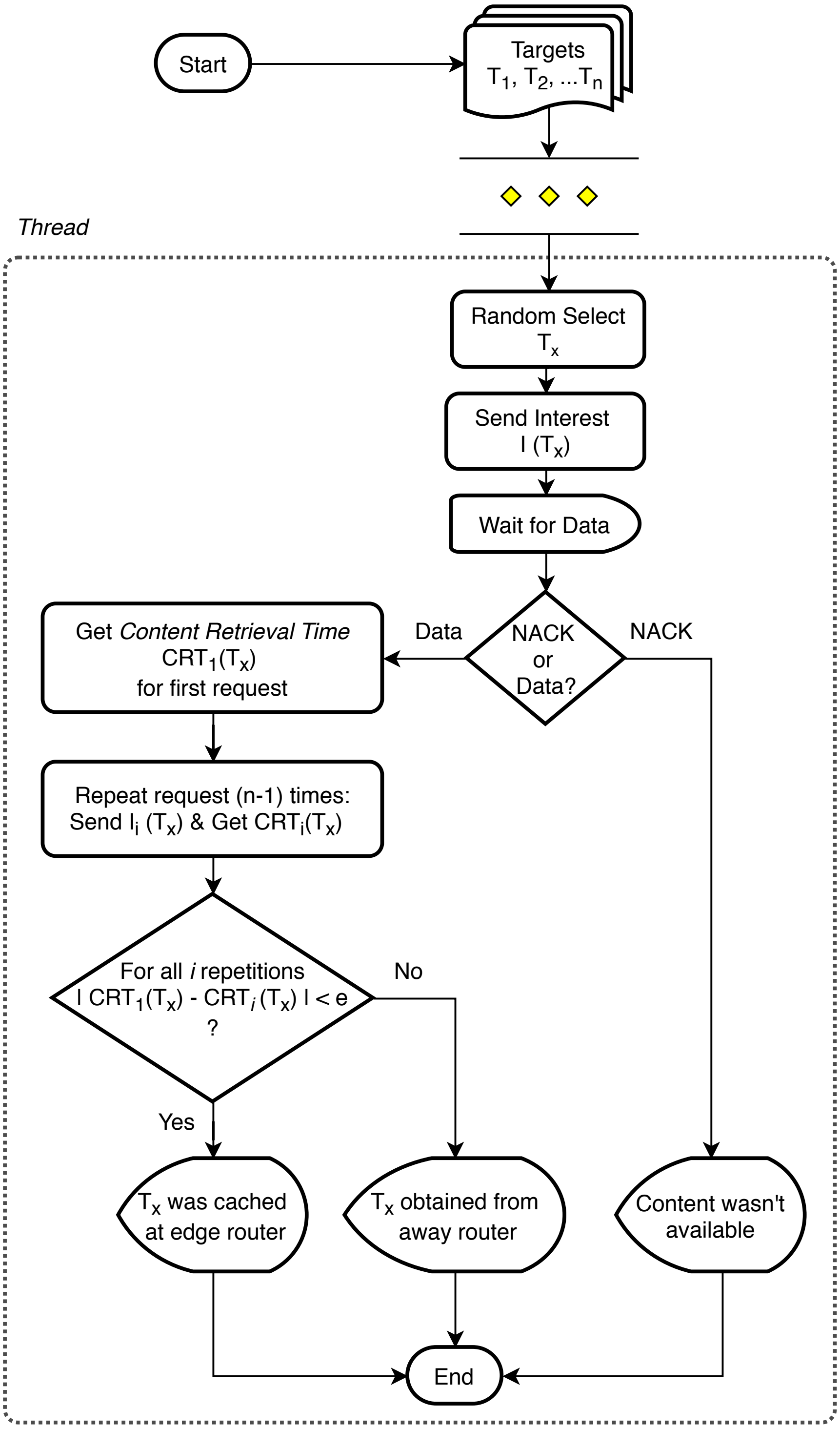


Figure 4.1: Brute-force side-channel timing attack flowchart.

If the target (*e.g.* streamed content or certificate) has not been produced by its producer, the NACK packet occurs with "content wasn't available" message.

The adversary may repeat the attack several times to distinguish the targets between cached and un-cached. In each repetition, the adversary retrieves different or same CRT values to conclude the target is cached by edge or neighbor/away routers. As shown in Figure 4.1, the adversary selects a target $(\mathrm{T_x})$ randomly from all targets $(\mathrm{T_n})$. In order to succeed in the attack, the adversary must repeat each target at least two times. In this algorithm, the attack repetition is defined as $\mathrm{n}-1$ which can be varied by side-channel attack design. Then a set of CRT values $(\mathrm{CRT_i})$ are obtained for repetition of a target $(\mathrm{T_x})$.

When the attack is finished, the adversary nodes can identify the target location by comparing their CRT. For instance, if the difference between the first CRT and all the others is small (less than $\epsilon$), the $\mathrm{CRT_i}$ presents the target $(\mathrm{T_x})$ has been recently cached by the by edge router otherwise it is cached from away routers. Also, the adversary can identify target' geographic distances in terms of hops by analyzing away routers $\mathrm{CRT_i}$ as also studied by [75].

## 4.2.2 Random Probing Function

Figure 4.2 illustrates the main differences between traditional and brute-force attack designs. In this example, each letter represents a content or its segment, **B** and **E** are cached contents, and the rest are un-cached ones.

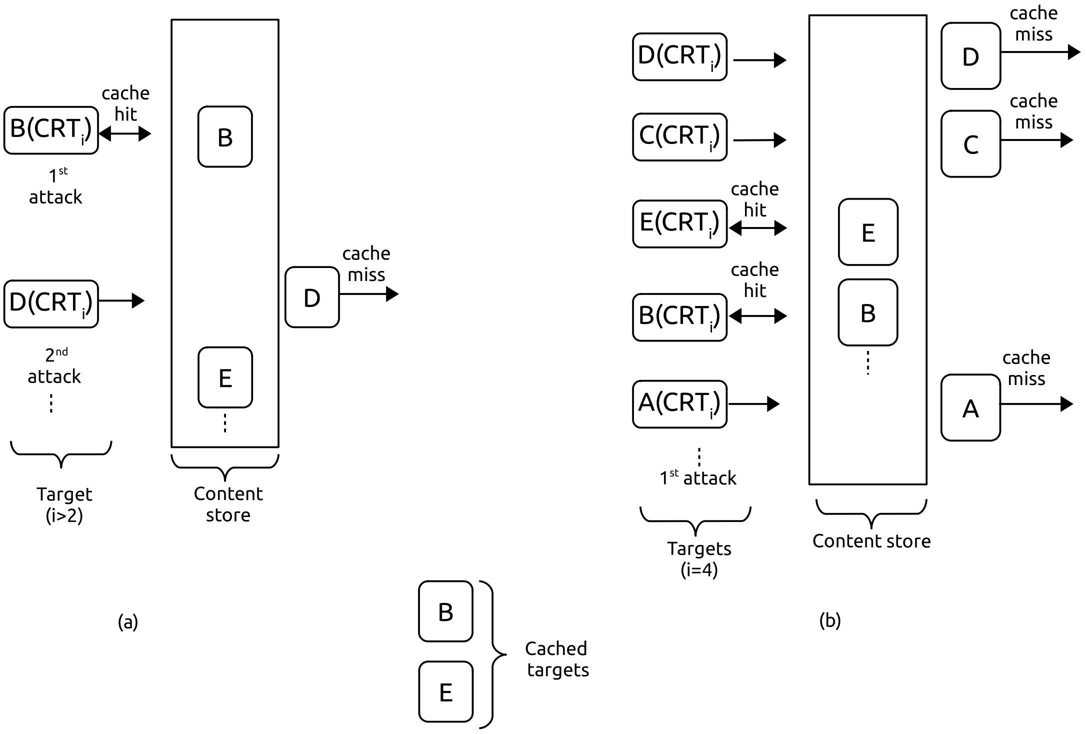


Figure 4.2: Comparison between traditional and brute-force attacks: (a) Traditional single target probing, (b) Randomized brute-force for multiple targets.

Figure 4.2a illustrates the adversary design in the traditional attack. This attack has a sequential design in which the adversaries may retrieve the targets one by one with at least two repetitions $(\mathrm{CRT}_2)$. Considering the importance of attack reliability, retrieving the targets one by one maybe not be the most efficient attack design. For instance, the adversary may succeed in the attack for **B**, but **E** could be discarded before the next attack time.

In the brute-force implementation, the adversary to improve the success ratio of the attack to retrieve targets randomly and in a short time (brute request). Figure 4.2b illustrates an example of the brute-force attack design with repetitions of four $(\mathrm{CRT}_4)$ for each target. Instead of probing [1] a single target, multiple targets can be identified in a brute-force attack.

The random probing increases the attack success ratio when compared with the sequential traditional design. For instance, as illustrated in Figure 4.2, the random probing boosts the attack for **E** and **B** and succeeds in the attack for both contents. However, this action wouldn't be possible in Figure 4.2a, because the **E** could be already discarded from the cache (depending on its cache policy).

Therefore, the attack can be configurable for multiple targets to succeed adversary may able to configure the multiple targets by the brute-force attack. Additionally, the adversary is also able to identify the approximate locations for the missed contents by comparing their CRTs.

### 4.2.3 Attack Scope on Applications

The attack success can be increased by brute-force with random probing functionality, compared to traditional attack design. Also, the scope of attack can be different on NDN applications. Therefore, in this work, the scope of brute-force attacks was classified based on two NDN applications: *i.* NDNtube presented as a YouTube-like user experience by serving dynamically generated video play-list and streaming (live-video) which serves the contents from one to many [61], and *ii.* VoNDN, a two-way voice conversation between peers and secured by a public key certificate scheme that exchanges the voice/video packets one-to-one [7].

**Streaming-like (NDNtube) application**. Figure 4.3 shows the consumer (Alice) requesting video (video-1234) from NDNtube producer and the corresponding answers with video frames (segments). In NDNtube, a video can be large enough to not be encapsulated by a single packet. Therefore, the media has to be split into data segments (*e.g.* video frames) to be delivered to the consumer. Also, the segments are cached by cache router to maintain the quality of the media.

In NDNtube, the cached video segments can be targeted by a brute-force attack. The scope of attack can be considered as monitoring the famous contents by also identifying their location. By brute force attack, the adversary can define multiple targets (various videos) to know what kind of video is being consumed by the router, by identifying the cached targets. Figure 4.3 also shows an example of an attack for a single

[1] definitions for "*retrieve*" and "*probing*" are used interchangeably in this thesis.

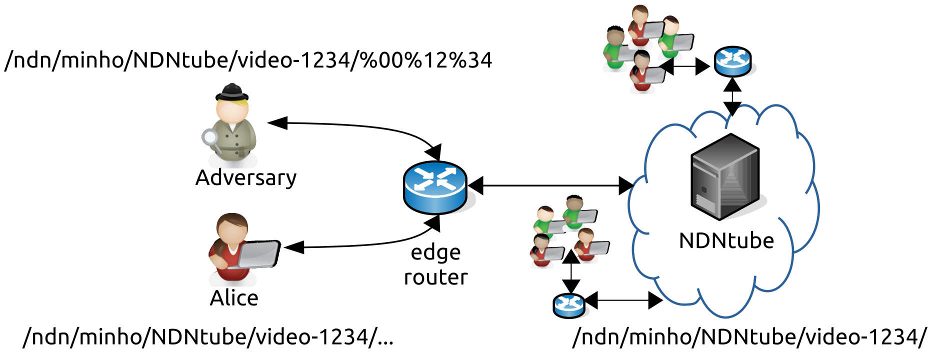

/ndn/minho/NDNtube/video-1234/%00%12%34
Adversary
Alice
edge
router
NDNtube
/ndn/minho/NDNtube/video-1234/...
/ndn/minho/NDNtube/video-1234/

Figure 4.3: NDNtube attack topology sample.

target (*/video-1234*). In this example, an adversary only probing the root of the target (*/video-1234*) from the router and it replied with a cached segment of that content (*/video-1234/%00%12%34*) to the adversary node. Note that, the scope is knowing the existence of the cached segment, not decoding the frame video.

**Trusted Voice over NDN application**. In VoNDN, the callee and the caller may exchange the certificates, these can be self-signed or produced by the certificate authority (CA). The certificates and media packets can be cached by routers to increase the availability of contents when required by callee and caller. However, the adversary may be targeted the cached certificates on trusted voice over NDN. Through this attack, the adversary may knowledge about where the callee or caller is located and also estimate the established conversation time.

On the other hand, targeted cached certificates may be present more than the locations to the adversary. In trusted based VoNDN, the attack is based on two adversary models: *i.* one able to identify the certificate at the edge router (Figure 4.4a), and *ii.* another able to identify the cached certificate from away routers (Figure 4.4b). Also, the attack can be configured for multiple targets.

The CRT side-channel timing attack has been based on two adversary models: *i.* one able to identify the certificates at the edge (closest) router (Figure 4.4a), and *ii.* another able to identify the cached certificate from away (more than one hop) routers (Figure 4.4b). When an adversary collects all possible CRT values, it can do three assumptions about where the certificate (target) has been cached.

First, the maximum CRT value shows that the certificate is not cached by any NDN routers except CA. Second, the minimum CRT value indicates the certificate has been recently cached by the edge router. Third, if the CRT is between a minimum and a maximum value, the adversary concludes that the content was cached by an away router. Note that, this router can be one hop (neighbor) or more than one hop away.

Additionally, more than one adversary can be used to determine the established call between Bob and Alice by attacking their certificates (Figure 4.4a). On the other hand, a single adversary may still determine Bob's certificate from away location (Figure 4.4b).

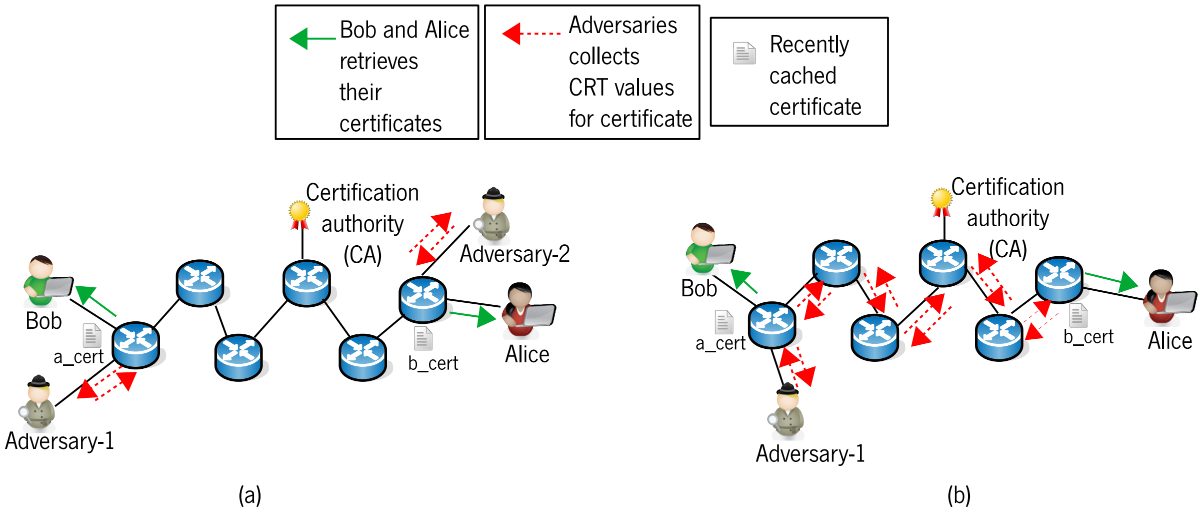

Bob and Alice retrieves their certificates
Adversaries collects CRT values for certificate
Recently cached certificate
Certification authority (CA)
Adversary-2
Bob
a_cert
Alice
b_cert
Adversary-1
(a)
Certification authority (CA)
Bob
a_cert
b_cert
Alice
Adversary-1
(b)

Figure 4.4: Side-channel timing attack on close and away targets: (a) Determine Bob's and Alice's certificate from the edge router location, (b) Determine Bob's certificate from away router location.

According to the cached certificate CRT comparisons, the scope of the trusted VoNDN attack can be summarized by the following determinations:

- Figure 4.4a illustrates that the adversaries (Adversary-1,-2) estimate the time of an established call between Alice and Bob from their edge routers by looking up their certificates (a_cert and b_cert).
- Figure 4.4a illustrates that the adversary (Adversary-1) identifies that who had a conversation with Bob recently by looking up to Alice's certificate (a_cert) from edge router. Also, the approximated the location of Alice by looking up to Bob's certificate (b_cert) from away location.
- Figure 4.4a and 4.4b are illustrated that the adversaries (Adversary-1,-2) identify where the call was established between Bob and Alice by comparing CRT responses of the certificates (a_cert and b_cert).

## 4.3 Detection and Defense Privacy model

Effectively, the side-channel timing attack can be mitigated by pre-configured countermeasure configurations, which were previously discussed (available cache and disabled methods Section 3.4). However, these methods may not be the most efficient ones, considering their configuration is static. This may reduce the certificate distribution efficiency for legitimate requests as described by [21], [53], and [18]. Thus, the countermeasures configurations can be considered as a trade-off between privacy and certificate distribution efficiency. To maintain the certificate distribution efficiency and protect the cached certificates this work proposes an approach called detection and defense (DaD).

The DaD is based on attack detection that can distinguish between legitimate and adversary faces. Through this adversary detection, the countermeasure method can be only applied to adversary detected face and legitimate requests can be preserved, without being affected by the available countermeasures. Also, the detection can be used to identify the severity of the attack. Then, different countermeasures can be applied to mitigate the attack. The DaD is based on three attack detection phases where available cache methods are applied in the first and second phases and disabled cache in the third phase.

The DaD identifies the attack in three phases as follows: *i.* **minor phase**, where the attack is detected in the first detection phase period window (TIME) and sets the adversary's face is configured with the available cache countermeasure for a time period, *ii.* **moderate phase**, where the attack persists in the second detection phase and sets the adversary's face is configured with a more effective available cache countermeasure compared to the first detection phase, and *iii.* **severe phase**, where the attack detected in the third detection phase, and the adversary's face is configured with the most effective countermeasure (disabled cache) to mitigate the attack.

The DaD framework can be summarized as follows:

**Detection methods and threshold calculations**: To apply the countermeasure methods to the adversary's face, the detection methods were presented. These are based on calculations as following: *i.* cache hit ratio (CHR), *ii.* hop counts, *iii.* shortest CRT value, and *iv.* statistically analyzed name prefix.

To detect the adversary's face a detection method threshold is needed. The DaD proposes a dynamic threshold value calculation using a pre-defined time. The calculations to detect the adversarial face will be presented in the next subsection.

During the attack, the CHR may indicate the adversary face in most of the NDN applications compared to other detection methods. Therefore, in this work, DaD detection is using the CHR to detect the adversary's face. However, the DaD can re-adapted to other detection methods to detect the adversary's face.

**Countermeasures impact and severity of attack**: Besides adversary detections, these can be used to obtain the severity of the attack and apply different countermeasures. Instead of applying a single countermeasure, DaD uses three different countermeasures (unpredictable delay, probabilistic caching, and no-cache) and these are applied according to the severity (minor, moderate, and severe) of the attack.

In this work, DaD is applied to NDNtube and VoNDN applications to detect the adversary's face. In these applications, each consumer node sends an interest packet within 100 interest/sec. by a constant bit rate (CBR). DaD considers the node as an adversary in case of face's CHR value is up to the CHR threshold. Also, this detection is different in used applications. For instance, DaD analyzing 50 packets in every 0.5s in NDNtube and 20 packets in every 0.2s in VoNDN applications. Thus, DaD distinguishes between legitimate and adversary nodes with detection methods which are discussed in the next subsection.

**DaD configuration on NDN applications**: The detection methods and threshold calculation time can be different depending on the NDN application. In DaD, more than one detection method can be used on privacy-sensitive applications such as VoNDN. Therefore, various detection methods can be adapted for the VoNDN and NDNtube applications.

From the simulation experiences, pre-defined CHR threshold values were used to detect the adversary faces by following values: 5% CHR in NDNtube (name-privacy) and 1% CHR in VoNDN (certificate-privacy) applications. Also, DaD collects these values every in 0.5 s in NDNtube and 0.2 s in VoNDN and apply countermeasures.

Next, the detection methods/thresholds such as cache hit ratio, hop counts, and CRT are presented.

### 4.3.1 Adversary Face Detection Methods

In this work, side-channel timing attack detection methods were surveyed. However, the number of related work in timing attacks on NDN is limited. For this reason, this work focused on to cache pollution attack detection method which has attack similarities with the side-channel timing attack in NDN [57]. In the cache pollution attack, the adversary may request the same content multiple times to disable the cache function of the router. Also, the adversary may create fake popularity for the contents to interfere with the distribution performance.

To detect an attack, the works [85] and [86] proposed that the cache hit ratio (CHR) can be used to detect the attack in cache pollution-related attacks. Due to the attack similarities between cache pollution and side-channel timing attack, the DaD attack detection is based on CHR calculations.

In DaD, the detection is face-based. The detection methods can detect the possible adversary face by getting metrics from NDN Forwarding Daemon (NFD). NFD is used as a network forwarder in the network layer. Therefore, the DaD can be used to distinguish between legitimate and adversary nodes to apply multiple countermeasures methods only to the adversary detected face.

In this work, three main detections methods were studied to mitigate the side-channel timing attack as following: *i.* cache hit ratio (CHR), *ii.* hop counts, and *iii.* content retrieval time (CRT).

**Cache hit ratio (CHR)**. During the attack, if the target has been requested previously and cached, the cache hit occurs in the next incoming request to the router. Therefore, the CHR can be used to identify the existence of the attack as proposed by [57]. On the other hand, in a side-channel attack, the adversary increases the CHR when the attack succeeds. Thus, the CHR can be used to obtain the following: *i.* the performance of the attack and *ii.* identify the face of the router that is being attacked.

Ideally, the cache hit ratio can be calculated periodically by an NDN application (*e.g.* NFD) to identify the face of the router that is being attacked in NDNtube or VoNDN. The average CHR of all edge routers is calculated by the total cache hits of each edge router using the following equation:

$$\mathrm{CHR} = \frac{\sum_{\mathrm{k=1}}^{\mathrm{n}} (\mathrm{total\_cache\_hits})_{\mathrm{k}}}{\mathrm{R}} * 100\% \tag{5}$$

where *n* is the total number of the edge routers in the network, and *R* is the total number of requests received by the edge routers, which is equal to the total number of cache hits plus the total number of cache misses.

The adversary may repeat the request to increase the success of the attack. Thus, the CHR can be used to measure the performance of the attack configuration. Our previous work studied CHR attack performance calculation for name privacy of the streaming NDN application (NDNtube) [87]. The results showed that the CHR increases if the adversary succeeds in the attack for previously cached contents.

Besides the attack performance calculation, the CHR can be used not only to detect an attack but also to identify the face of the router that is being attacked. For instance, the CHR can be used to identify the adversary in an attack detection for cache pollution [86] [85]. Also, the works [57], [88], and [89] claims that the side-channel timing attack can be identified by cache hit and misses.

In DaD, the attack detection is based on CHR calculation per face for VoNDN application. To detect the adversary's face, the DaD uses the CHR threshold value. If a face's CHR value calculated is higher than the threshold, that face of the router is considered an adversary. Thus, the DaD distinguishes between legitimate and adversary nodes to apply countermeasures.

**CHR threshold calculation**. In DaD, the CHR threshold parameter is used to identify the face of the router that is being attacked. This parameter is calculated as follows. A set of *m* requests is collected regularly during $\Delta\mathrm{T}$ seconds. The total number of cache hits is calculated for the new set of requests, which we consider to be the $\mathrm{i^{th}}$ collected set. So, the average CHR of this new set ($\mathrm{chr_i}$) is calculated by the following equation:

$$\mathrm{chr_i} = \frac{\sum_{\mathrm{k}=1}^{\mathrm{m}} \mathrm{CH_k}}{\mathrm{m}} \tag{6}$$

where $\mathrm{CH_k}$ represents the cache hit of the $\mathrm{k^{th}}$ request in the new set. The $\mathrm{CH_k}$ is one if the $\mathrm{k^{th}}$ request gets a cache hit and zero in case of a cache miss. Then, the new global average $\mathrm{CHR_j}$ is computed by the following weighted moving average equation:

$$\mathrm{CHR_j} = (\alpha \times \mathrm{CHR_{j-1}}) + (1 - \alpha) \times (\mathrm{chr_i}) \tag{7}$$

where $\mathrm{CHR_{j-1}}$ represents the last CHR value, $\mathrm{chr_i}$ is the new value calculated by Eq. 6, and $\alpha$ is a weight factor between 0 and 1. The $\mathrm{CHR_j}$ is very sensible to the new $\mathrm{chr_i}$ value if $\alpha$ is close to 0, and little sensible if $\alpha$ is close to 1. In DaD, $\alpha$ should be chosen close to 0, because an attack increases the $\mathrm{CHR_i}$ when it is established, and so the system can detect it quickly. For this reason, $\alpha$ was set close to 0 in our experimental NDN-testbed scenario. The router is considered under attack if $\mathrm{CHR_j}$ is higher than the threshold CHR. The CHR thresholds were identified as follows: 5% CHR in NDNtube (name-privacy) and 1% CHR in VoNDN (certificate-privacy) applications. Note that, these thresholds can be defined manually, or dynamically by an algorithm based, for instance, on machine learning techniques.

**Hop counts**. When an interest packet is sent, the hop count increments by one whenever it passes through an intermediate router. For instance, the minimum hop count occurs when the content (*e.g.* media, voice, certificate) is cached by the edge router, as illustrated in Figure 4.5.

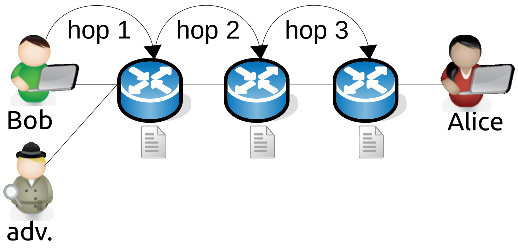


Figure 4.5: Attack detection by hop counts.

When the content is cached by the edge router and re-requested by a consumer (adversary included) the minimum hop count occurs for the corresponding face. Otherwise, a higher hop count occurs between $1 < \mathrm{hop\_count} \leq \mathrm{maximum}$. During the attack, the hop count variance may change dramatically for the adversary face [86] [90]. This variance can be motivated by the request repetitions.

Ideally, the hop count of the adversary's face may be calculated with a minimum in streamed contents. The reason is that these contents are cached by multiple routers (broadcast or multicast). Thus, the adversary's minimum hop counts can be significantly higher than the legitimate consumer which can be used to distinguish between adversary and legitimate faces.

On the other hand, the other hop counts may be used to detect the adversary's face in certificate-based attacks. For instance, the certificates may not be cached as streamed contents (NDNtube) by the edge routers. Thus, the hop count of the adversary's face may be calculated higher than the legitimate nodes which can be used to distinguish between adversary and legitimate faces.

**Shortest CRT value**. The shortest CRT value identifies the cached target by the edge router. However, also retrieving other un-cached contents, the adversary node may reach the highest CRT values in a short time. Ideally, this increases the CRT frequency for the targets and the threshold can be used to detect the attacked router so the adversary's face. For instance, an adversary can be detected if any face's CRT is lower than its threshold value (*e.g.* $\mathrm{CRT}_{\mathrm{n}} < \mathrm{CRT}_{\mathrm{threshold}}$) which is based on CRT calculations as previously presented (Subsection 3.2.1).

**Name prefix**. In the brute-force attack, the attack is based on multiple targets. This changes the pattern of the requested contents and can be used to identify the adversary face on the router [65], [74]. For instance, when an adversary targeted the certificates (*e.g. /domain/vondn/KEY/cert-target*), a number of certificate requests can be increased comparing the other requests (*e.g. /vondn/media*). Therefore, an adversary face can be detected by statistically analyzing the number of requests for the name prefixes.

On the other hand, the adversary may be targeting a specific namespace rather than other namespaces that belong to the same domain. In this brute-force attack scenario, the amount of request may be increased for particular namespaces (*e.g. /uminho/eng/di*) compared to other name-spaces (e.g. */uminho/eng/dps*). This can be identified statistically by analyzing the requests of each namespace.

**Discussion**. In this work, four detection methods were studied to detect an adversary face in NDNtube (streaming) and trusted-VoNDN applications. These methods can also correlate with each other. For instance, an adversary face can have an increased CHR value and name prefix requests while its hop count and CRT are calculated as a minimum. If these four metrics are available from the applications, the combination of all metrics maybe provides almost "*perfect privacy*" to the NDN applications.

Note that, the CHR value is generically available for all caching NDN applications and this makes it an effective detection method. On the other hand, privacy concerns may be different in NDN applications. For instance, an adversary may identify the location of the callee or the caller by identifying cached certificate CRTs on a trusted-VoNDN application. Ideally, in this attack, an adversary requests a number of certificates, and legitimate nodes usually request a single certificate in a short time. Because of this difference, CHR and name-prefix detections can be used to detect an adversary effortlessly compared to the NDNtube application.

In NDNtube, an adversary may use the attack for monitoring. In this case, a streamed content is cached by corresponding routers, and the adversary uses CRT responses from the edge routers. Since an adversary attacks multiple targets, several hop counts maybe occurred for the corresponding face. For instance, some targets may be cached and others are cached by a neighbor or away routers. Thus, a hop count information can be a metric to detect the adversary node in streaming-like NDN applications such as NDNtube.

### 4.3.2 Countermeasures Impact and Severity of Attack

The face of the router that is being attacked can be identified by the detection methods to apply countermeasures. However, configuring the router with static countermeasures may not be the most appropriate approach, considering that each of the countermeasures configurations effects can be different on mitigation and distribution efficiency.

In DaD, an adversary detected face can be set with different countermeasures. When the adversary is detected by CHR, the countermeasure is only applied to the possible identified face. The side-channel adversary detection by CHR can be used: *i.* to configure a countermeasure when the adversary is detected, and *ii.* to determine the severity of the attack which can be used to set different countermeasures.

**Countermeasure impact**. The DaD is based on three naïve countermeasures to apply available cache configuration by the following:

*i.* unpredictable delay: The attack can be mitigated by uniform (fixed), random distribution, and unpredictable delays. However, because the unpredictable delay may be the most challenging to solve by the adversary that is compared to other delay distributions (fixed or random). This is explained by the complexity of the cryptographic function of unpredictable delays.

The unpredictable delay is calculated by a hash function [69], [88]. This function can be expressed by *h(u, k)* mod $\mathrm{m}_{\max}$ = *h(k)*, where *k* is a cryptographic integer hash code from the key that generates unpredictable delay *h(k)*. Then, this delay is added to the adversary detected face to challenge the adversary CRT calculations.

*ii.* probabilistically caching: The edge router can cache the contents by probabilistically [70]. This can be used to mitigate the brute-force attack. The DaD offers the contents that can be cached by *p*=10% probabilistically selection to the adversary detected face. This means that the edge router is randomly selecting (based on probabilistic function) the content from its cache and answering (random distribution) to the adversary detected face instead of answering directly.

Note that, in case of a detected face is directed to multiple addresses (devices), the legitimate consumers also are affected by the probabilistic cache. In this case, the edge router can probabilistically cache for all faces because it can be a challenge to distinguish between adversary and legitimate faces.

*iii.* no caching: This completely disables the cache and the adversary cannot succeed in any attack.

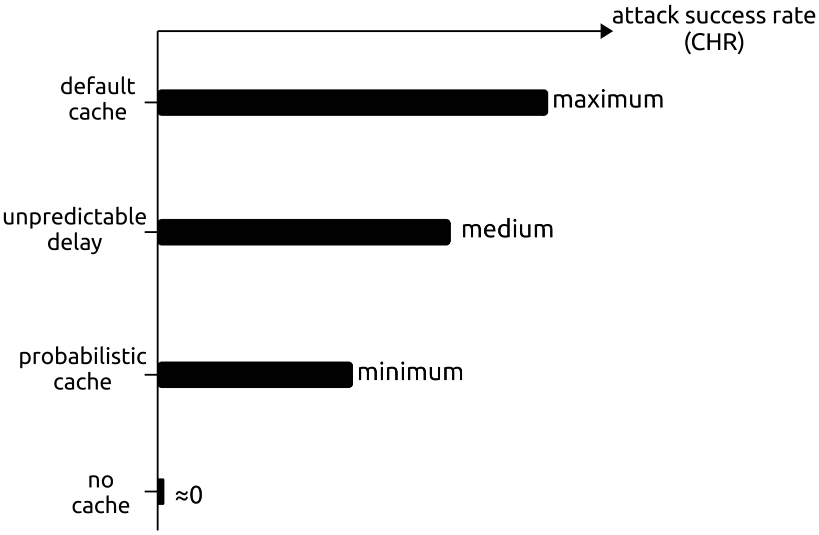


Figure 4.6: A qualitative analysis of the countermeasures impact on the attack success.

Figure 4.6 compares qualitatively the success of an attack based on default and countermeasures configurations efficiency to mitigate the attack. In this graph, the success of the attack is analyzed based on the cache hit ratio (CHR). For instance, the attack cannot have any success rate in no-cache configuration because it completely disables the cache.

On the other hand, the additional unpredictable delay can be considered to have a better distribution efficiency than the probabilistic cache, because the contents can be already cached by the router in the delay configuration. In this case, the adversary may determine that additional delay by multiple trails of attacks. However, the cache cannot be fully loaded by a randomized distribution configuration. Therefore, in an ideal situation of attack, the unpredictable delay configuration can be considered less effective compared to a randomized distribution configuration.

Note that configuring the router with a high unpredictable delay or low probabilistic rate of caching may affect severely the content distribution efficiency.

**Determine the severity of attack**. In DaD, the CHR can be also used to identify the severity of the attack. For instance, if the attack is detected in a period (TIME) and continued in the next detection states, the attack can be considered severe. Note that, the detection period can be tuned by application (NDNtube or VoNDN) configurations. For instance, the adversary was not detected in a higher attack check time because it was already completed the attack. On the other hand, a shorter check time may slow the process of the router. Therefore, through several simulation experiences, an optimum DaD period check attack time was defined as 0.5s for NDNtube and 0.2s for VoNDN applications.

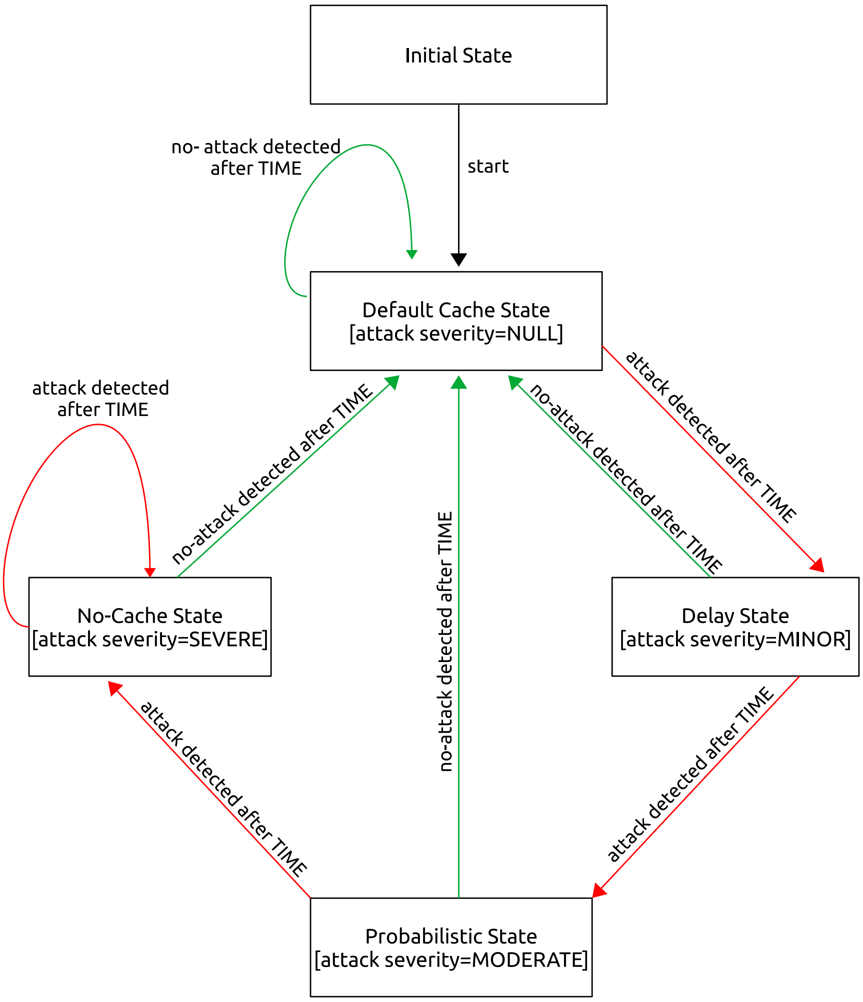

Initial State
start
no- attack detected after TIME
Default Cache State [attack severity=NULL]
attack detected after TIME
no-attack detected after TIME
no-attack detected after TIME
no-attack detected after TIME
attack detected after TIME
No-Cache State [attack severity=SEVERE]
Delay State [attack severity=MINOR]
attack detected after TIME
attack detected after TIME
Probabilistic State [attack severity=MODERATE]

Figure 4.7: Attack states (phases) and applied countermeasures.

According to the attack severity, different countermeasures methods can be applied to mitigate the attack and maintain the content distribution, as illustrated in Figure 4.7. For instance, the unpredictable delay can be applied in the first detection state (*minor*), the probabilistic caching in the second detection state (*moderate*), and no-cache in the last state (*severe*) to mitigate the effect of the attack. Also, when a detected attack is severe, the no-cache countermeasure is applied while the attack persists. If the adversary withdraws the attack in any detection state, the router is set to the default state (*e.g.* default LRU caching).

In this work, the attack severity is obtained by the CHR threshold. These values were used to detect the adversary faces by following values: 5% CHR in NDNtube (name-privacy) and 1% CHR in VoNDN (certificate-privacy) applications. Also, DaD collects these values every in 0.5 s in NDNtube and 0.2 s in VoNDN. The countermeasures phases (minor, moderate, and severe) are applied for 3 s in NDNtube and 2 s in VoNDN respectively.

In NDNtube, DaD analyzing ≈50 packets (100 packets/0.5s) every 0.5 seconds to define the severity of the attack. If a face is detected as an adversary DaD sets the attack states (minor, moderate, and severe) for 3 seconds respectively.

In VoNDN, DaD is analyzing ≈20 packets every 0.2 seconds for all faces. Like NDNtube, DaD sets the minor, moderate, and severe states (phases) for 2 seconds depending on the severity of the attack.

Next, the DaD algorithm and its configuration for NDNtube and VoNDN are presented.

**Detection and Defense (DaD) algorithm**. DaD can be designed for any NDN application to maintain privacy and content distribution. In fact, instead of setting the router with a static countermeasure method, DaD applies dynamically three countermeasures when the router is under attack.

In this work, the DaD algorithm was designed for both name and certificate privacy in a trusted VoNDN and streaming NDNtube applications. Ideally, the algorithm can be implemented in the application layer to protect cached streamed content and certificates. This design is based on the CHR threshold to detect the adversary face during the attack. DaD checks the existence of an attack and applies the countermeasures every TIME seconds.

If the CHR is above the CHR threshold, the DaD identifies it as an adversary's face and defines the severity of attack as minor, moderate, or severe. To protect the content and certificate privacy in NDNtube and VoNDN, DaD was designed based on the following three countermeasures: unpredictable delay, probabilistic cache, and no-cache.

The DaD algorithm is presented in Figure 4.8. The algorithm uses two attack states (attack detected, no-attack detected), determined by the attack detection method, which is in this case is based on CHR metrics. *i.* **no attack detected**. The default cache replacement policy is applied to all faces, which may be, for instance, LRU. *ii.* **attack detected**. The attack is detected by the detection threshold in the router. The DaD uses, the cache hit ratio to detect the attack. However, other detections mechanisms, such as CRT and name-prefix analyzes can be adapted to the algorithm.

Depending on the severity of the attack, three countermeasures can be applied by DaD to the detected adversary face. Let us consider the application runs at the default phase (no attack detected) and the DaD checks the existence of the attack every TIME seconds. Let us consider that TIME is equal to two seconds for the VoNDN application. If the attack is detected after the check time, the router switches from the default phase to the attack detected phase. In this case, the DaD identifies the attack as minor for 2 seconds and applies a delay phase to the adversary's face. If the attack continues in the next attack check, then DaD considers the attack as moderate and sets the router for the random phase for another 2 seconds to the adversary's face. In the next attack check, if the attack persists, DaD sets the adversary's face of the attacked router for no-cache and keeps always in this state while the attack exists in the next attack checks. If the attack is withdrawn in any phase, then the router returns to the default phase and the whole process restarts again.

Table 4.1: DaD algorithm description and parameters.

| | | |
|---|---|---|
| **inputs** | TIME | cacheHitTreshold |

| | |
|---|---|
| **auxiliary processes** | getCacheHitRatio()<br>apply_defaultPolicy()<br>apply_Delay()<br>apply_Random()<br>apply_noCache() |

| | | |
|---|---|---|
| **attack state** | ATTACK_DETECTED | delayPhase |
| | | randomPhase |
| | | noCachePhase |
| | NO_ATTACK_DETECTED | defaultPhase |

Table 4.1 shows the main parameters and the attack phases used by the DaD algorithm. The TIME and the cacheHitTreshold are pre-defined parameters used by this algorithm. TIME is the period used to detect the existence of an attack per face. So, it is the check attack time, which means that every TIME seconds the router is checked for an attack for each face. The router is considered under attack when the cache hit ratio (CHR) is higher than the cacheHitTreshold for a certain face. The auxiliary getCacheHitRatio() process is used to obtain the CHR from each face of the router where DaD is running. If the attack is detected (ATTACK_DETECTED), then distinct countermeasures are applied according to the severity of the attack during the pre-defined time (TIME).

So, to recapitulate the DaD procedure for contents (streamed or certificate), when an attack is firstly detected, the router enters the delay phase (delayPhase) and keeps in this phase until the next attack check, which will occur TIME seconds later. During the delay phase, all contents are sent through the detected face with an additional unpredictable delay set by the apply_Delay() process. If the attack persists in the second attack check, then the attack detected face enters the random phase (randomPhase). During this phase, the cached contents are selected from the router's cache with the apply_Random() process to be distributed to the adversary detected face. This random distribution is based on a probabilistic function.

If the attack persists in the third detection period check, then the attack detected face enters the no-cache phase (noCachePhase) and stays in this phase while the attack persists. During this phase, no contents are stored in the cache of the detected face. If the attack is withdrawn in any phase of the ATTACK_DETECTED condition, the router switches to the default phase (defaultPhase).

If the CHR is not above the cacheHitTreshold, then no attack is detected (NO_ATTACK_DETECTED), which establishes the default phase by setting the apply_defaultPolicy() process. In this phase, DaD applies the default caching policy to the router.

The DaD uses the CHR detection metric in privacy-sensitive applications (*e.g.* trusted VoNDN) to identify the adversary node. However, this metric depends on the type of application. Since it is not possible to define a *priori* a threshold for all applications, this may require that the attack check period (TIME) be adjusted automatically by the application. In this way, the DaD algorithm could be adapted to the NDNtube and VoNDN applications, as discussed next.

### 4.3.3 DaD Configuration on Applications

The caching strategy and the scope of the attack can be various on VoNDN and NDNtube applications. Thus, the detection methods can be also different on applications to distinguish between adversary and legitimate faces. In this work, the privacy-oriented issues were discussed on the NDNtube and VoNDN applications. Ideally, the DaD can be tuned according to the NDNtube and VoNDN applications.

NDNtube and VoNDN are considered large-scale real-time applications. In these applications, the backbone routers' CHR can be calculated high because of the edge routers' requests. Therefore, the cache hit ratio detection is only feasible in the edge router's in the NDNtube and VoNDN applications by naïve DaD design.

**DaD in NDNtube application**. NDNtube can distribute pre-recorded video by split frames to the cache. The adversary may deploy a brute-force attack for multiple targets to determine the popularity of video. This attack can be also considered as monitoring and censorship in the content locations.

NDNtube is a large-scale application and a single detection method (cache hit ratio) may be used to obtain the adversary face and severity of the attack on the edge router. In this detection, the adversary's face is detected if it overcomes the thresholds in 0.5 seconds. Then, multiple countermeasures can be applied depending on the severity of the attack: *i.* unpredictable delay, *ii.* probabilistic cache, and *iii.* no-cache.

In NDNtube, DaD continuously gets the CHR from each face and analyzing ≈50 packets (100 packets/0.5s) every 0.5 seconds. If a face's CHR is over 5%, DaD sets the minor attack phase for 3 seconds and keeps detecting the attacks every 0.5 seconds. If the attack continues, DaD sets the moderate phase for another 3 seconds and checks the attack every 0.5 seconds. If the attack continues, DaD sets a severe phase and keeps it while the attack is detected.

**DaD in trusted VoNDN application**. The adversary may identify the consumers' private information, such as approximated locations through a targeted certificate in VoNDN. In trusted VoNDN application, the certificates provide integrity for callee and caller. The certificate can be issued by CA or self-signed. However, the cached certificate may provide such pieces of information when it is targeted by the adversary, namely: *i.* the name of callee/caller, *ii.* the location of callee/caller, and *iii.* the approximate time of the conversation.

In VoNDN, the adversary's face is detected if it overcomes the thresholds in 0.2 seconds. Then, multiple countermeasures can be applied by depending on the severity of the attack: *i.* unpredictable delay, *ii.* probabilistic cache, and *iii.* no-cache. In this design, the CHR threshold value can be used to obtain the adversary's face and to identify the severity of the attack. Also, the DaD allows updating the threshold value in each pre-defined time to make decisions about the adversary and countermeasure.

In VoNDN, DaD considers adversary face if a face's CHR up to 1%. In this application, DaD is analyzing ≈20 packets every 0.2 seconds for all faces. As presented previously, DaD sets the minor, moderate, and severe phases depending on CHR thresholds these calculated every 0.2 seconds.

## 4.4 Summary

This chapter presented the contributions of this work. First, an attack design called brute-force (burst-like) was presented that increases the success of the attack comparing to the traditional attack design. In this attack model, multiple targets can be defined by an adversary and probing the targets using a random function. Also, the attack scope is analyzed for NDNtube and VoNDN applications.

To mitigate such brute-force and traditional attack models, the countermeasure methods such as unpredictable delay, randomized distribution (based on probabilistic function), and no-cache are used. The countermeasure affects the attack effects and the content distribution performance was also illustrated. The attack detection methods (CHR, hop count, name-prefix, and CRT) were presented to detect the adversary's face and the severity of the attack.

A combined approach with detection and countermeasure method was presented to mitigate the side-channel timing attack efficiently in NDN applications. Instead of a statically configured router with a countermeasure, this work approach called detection and defense (DaD) applies the multiple countermeasure methods only in the attacked router. The various countermeasures methods can be applied from less to more effective ones depending on the severity of an attack. Through that, legitimate requests can be also protected because no countermeasure was applied to un-attacked routers.

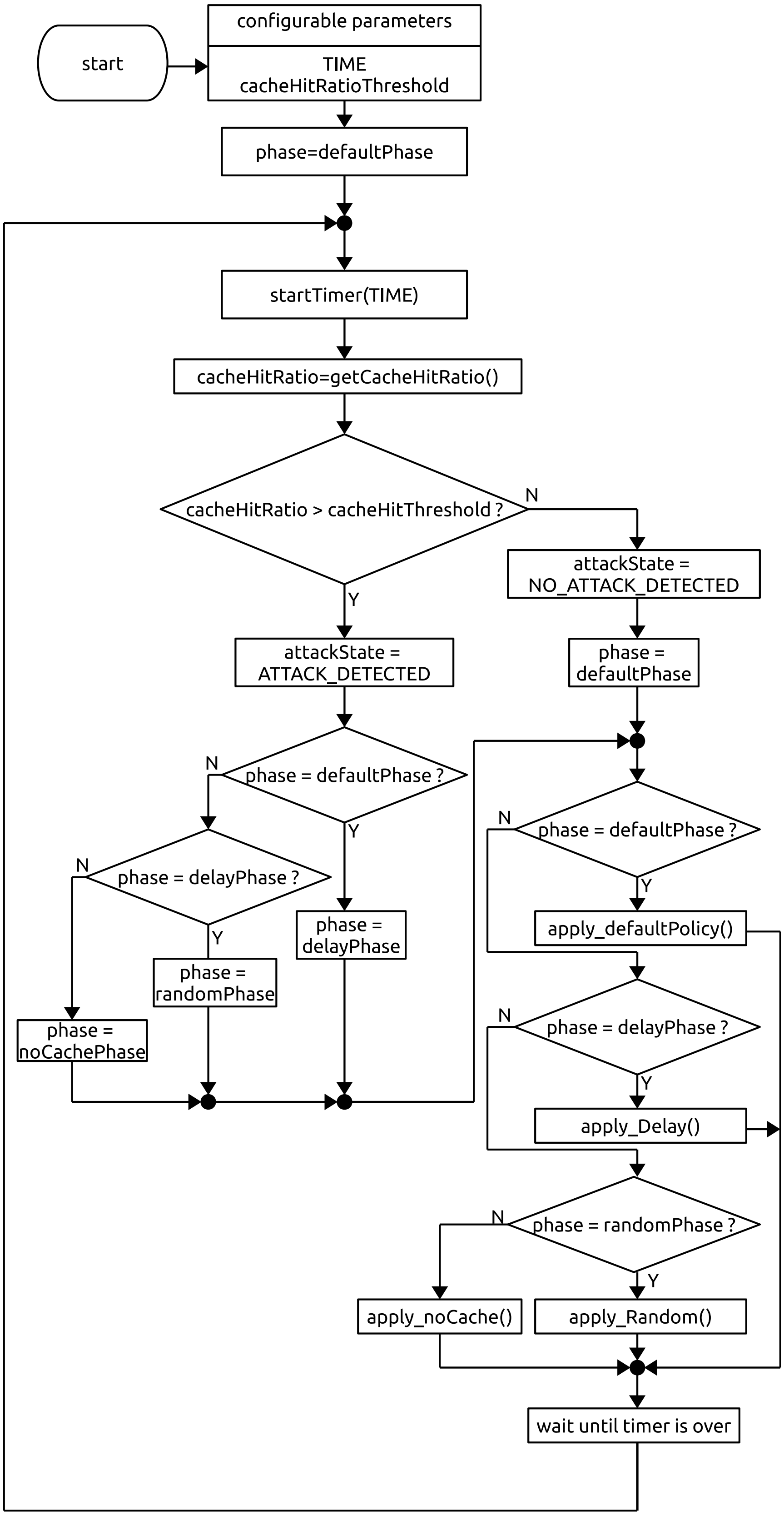


Figure 4.8: Detection and Defense (DaD) flowchart algorithm.

# 5

# EXPERIMENTAL FRAMEWORK AND IMPLEMENTATION

In this chapter, a description of the framework used for the implementation is presented, followed by an overview of the proposed modules designed and implemented to create the testing scenario. Also, the used simulator instruments and their usage were discussed to develop this work implementation.

## 5.1 Context

The NDN research has two main experimental platforms to test and improve the NDN instruments, these are *i.* NDN simulator (ndnSIM) and *ii.* NDN-testbed. Both instruments were developed to update and upgrade the NDN paradigm libraries (ndn-cxx) and NDN daemons.

In this work, all experiment scenarios and supporting modules were coded on the NDN simulator (ndnSIM). Also, the ndnSIM components such as consumer, producer, content store policies, NDN forwarding daemon (NFD), and forwarding strategies are presented. They were used to build attack implementation scenarios. The latest version 2.6 (Jan.2019) [12] of ndnSIM with and integration of NFD 0.6.5 [33] was used in the implementation discussed in this chapter.

## 5.2 Named Data Networking Simulator

The ndnSIM is an open-source simulator platform that is used to conduct the research needs of the NDN architecture. The simulator is based on the ns-3 (network simulator 3) [91] simulator platform which is also used to script the C++ based network simulations.

The ndnSIM can be installed in Linux environments (primarily Ubuntu), either natively or in a Virtual Machine. In this work, the ndnSIM 2.5 was installed on Ubuntu 16.04. The simulator also integrates the NFD version 0.6.5, which is presented in the next section.

Table 5.1 shows the configurations of ndnSIM simulation setups. In this work, two machines were used because of scenarios script for long-term periods which may take days to complete. The VM was

Table 5.1: Linux machines setup.

| **Machines** | **Linux** | **ndnSIM version** | **NFD version** | **RAM** | **CPU** |
|---|---|---|---|---|---|
| Virtual Machine | Ubuntu 16.04 | 2.6 | 0.6.5 | 4gb | 2 cores (psychical)<br>4 threads (virtual cores) |
| Native Machine | Ubuntu 16.04 | 2.6 | 0.6.5 | 16gb | 4 cores (psychical)<br>8 threads (virtual cores) |

used mainly for testing purposes and to run small simulations, while the Native OS was used to run larger simulations.

### 5.2.1 Network Simulator 3

Network simulator 3 (ns-3) [92] is a discrete-event network simulator targeted primarily for research and educational use. It has been developed to provide an extensible network simulation platform to be used by researchers and developers for experimental network application simulations. The ns-3 is based on network models, that aim to analyze how data packet performs and provides a simulation engine to conduct the simulation experiments.

Unlike its predecessor ns-2 [9], the ns-3 may support the existing Internet protocols (IP) and other networks (non-IP) such as NDN. NS-3 is not backward compatible with ns-2. It is a completely new simulator mainly written in C++. Also, ns-3 can be extended by external software libraries and run simulations scripted in Python or C++. Currently, the ns-3 is primarily used on Linux systems, but it can also run on FreeBSD, macOS, and also on Windows systems using the Cygwin platform [9].

Table 5.2: The ns-3 features.

| **Components** | **ns-3** |
|---|---|
| Languages | C++ and Python |
| Packets | The buffer answers realistic to the stream |
| Simulations | Provides a lower base level of abstraction |
| Maintenance | Actively maintained (email support) |
| Visualization | Python |
| Network layer | IP and non-IP architectures |

Table 5.2 shows the main features of ns-3. Through the open-source C++ and Python supported libraries, the ns-3 may provide long-term maintenance support in IP and non-IP networks. Also, the ns-3 performs the most realistic simulations e.g. buffer memories can send the packets in bits as real streaming applications. This also engages with a lower base level of abstraction such a realistic consumer and producer behavior as described by the ns-3 tutorial book [91].

ns-3 is built with a build tool called waf which is directly configured and builds the ns-3. Figure 5.1 illustrates how to add module (IP or non-IP) into the wscipt which is used to identify new protocol addition. Then, this new protocol can be compiled by ./waf which is a python-based build tool. After executing the ./waf command, the simulation calls that protocol with the packet header and timer to structure the simulation experiment.

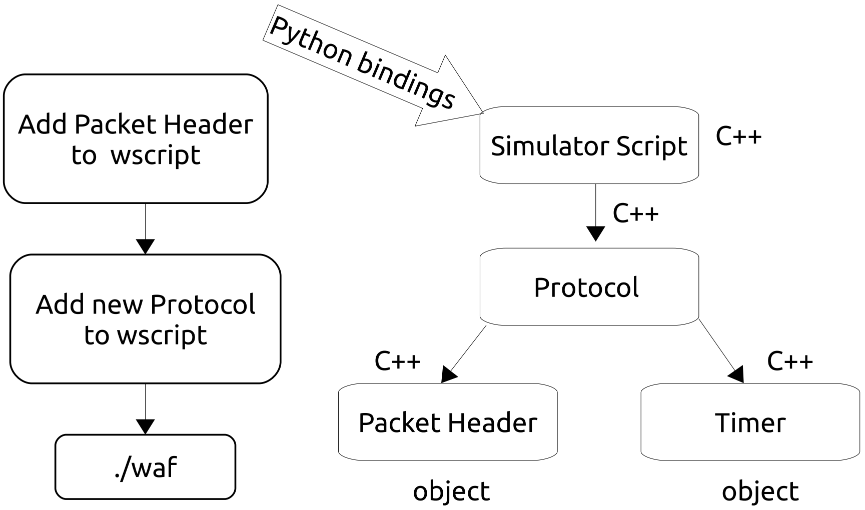


Figure 5.1: ns-3 core structure (adapted from [9]).

## 5.2.2 ndnSIM Helpers

To meet with NDN architecture and its application need, the ndnSIM was created as an ns-3 module that implements NDN communication models. So far, the ndnSIM has two versions: ndnSIM 1.0 ([10]) and ndnSIM 2.x ([11], [12]). The main difference between these two versions, the ndnSIM 2.x is supported by NFD integration which ndnSIM 1.0 does not have.

ndnSIM is implemented as an additional network-layer protocol that can be run on top of any link-layer (L2) such as PPP, MAC, VLAN, WIFI, CSMA, etc., network-layers(L3) such as IPv4, IPv6 and NAT, and transport-layers (L4) such as TCP and UDP. With this approach, the IP networks can be also adapted to the ndnSIM.

The key features of ndnSIM can be summarized as follows:

- Supports to NDN packet formats such as interest, data, NACK, and certificate.
- Includes the NDN C++ libraries (ndn-cxx).
- NFD integration and management.

ndnSIM is implemented with different C++ libraries to simulate NDN entities such as FIB, PIT, CS, and Faces. These modular structures enable each module for modification or replacement without affecting others. Also, the ndn-cxx libraries extend for NDN needs by new releases.

ndn-cxx is a C++ library for NDN primitives that can be used for NDN applications. The ndn-cxx library is used to support the following simulation components: NDN Forwarding Daemon (NFD), NDN Link-State Routing Protocol (NLSR), Next-generation NDN repository (repo-ng), multi-user NDN chat application (ChronoChat), Sync library for multi-user real-time applications (ChronoSync), NDN essential tools (ndn-tools) and traffic generator for NDN (ndn-traffic-generator) as described by [93].

Table 5.3: ndnSIM code guide.

| location | description |
|---|---|
| model/ | NDN base: L3Protocol, faces (Face, NetDeviceTransport, AppLinkService) |
| NFD/ | NDN Forwarding Daemon (NFD) source code |
| ndn-cxx/ | ndn-cxx library source code |
| apps/ | Producer and consumer applications (ConsumerCbr, ConsumerWindow, ConsumerBatches, ConsumerZipfMandelbrot). |
| utils/ | Helpers for data structure and topology reader |
| helper/ | Contain several example scenarios |

**ndnSIM code guide**. Table 5.3 shows the location of ndn components in ndnSIM. Depending on the need of the simulation scenario, the components can be used to create simulations.

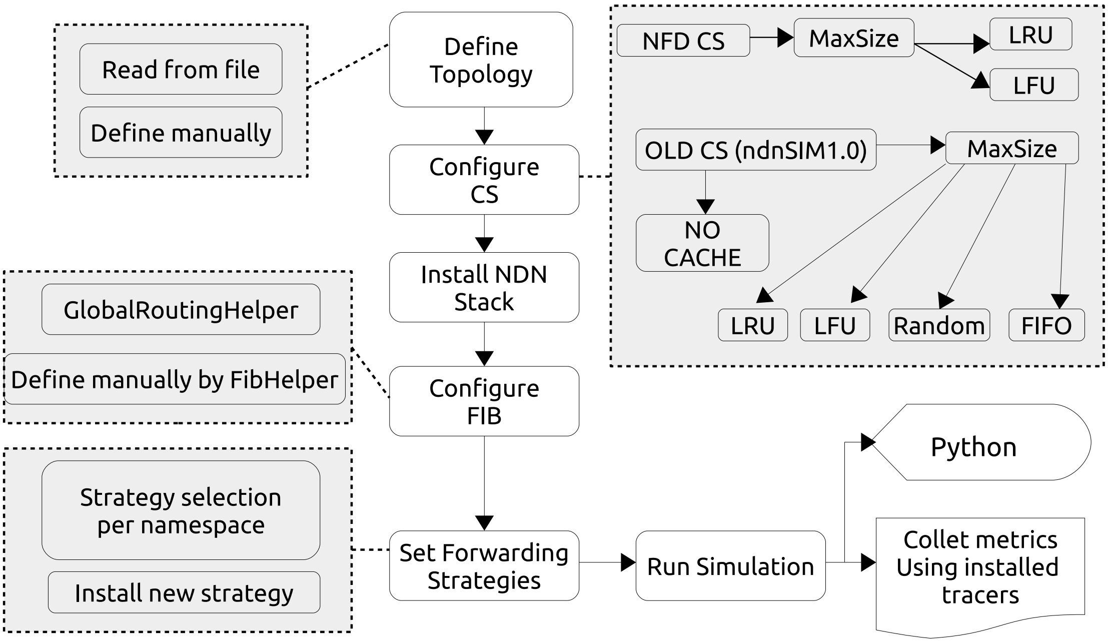


Figure 5.2: ndnSIM work-flow.

Figure 5.2 shows the workflow on ndnSIM. The real topology datasets can be read by the simulator to get realistic simulation results. Also, the simulator provides several content store implementations to be used according to the needs of the scenario application. These implementations can be installed to nodes by NDN stack helper. The forwarding strategies may require knowledge of FIB information that can be managed by configurable “global router helper” and “manual FIB” helpers.

Also, different forwarding strategies can be configured by the following: *i.* best route: where packets are routed to the best path of nodes, *ii.* multicast: where packets are routed to a group of nodes, and *iii.* broadcast: where packets are routed to all nodes. As like NDN, the forwarding strategies are associated with name-spaces and can be installed distinctively to each node.

The ndnSIM helpers are required to perform detailed tracing and NDN network traffic flow of the nodes. The scenario that was created in this work uses some of the ndnSIM helpers to meet the application needs. The used ndnSIM helpers for this work experiment scenario are described as follows:

**ns3::ndn::ConsumerCbr** The scenario consumer(s) sends the interest packets within `ConsumerCbr` pattern. In the ConsumerCbr application, the interest name prefixes can be configurable with a sequence number. Also, the interest frequency can be defined by various rates such as uniform (0, 1/frequency), random, and exponential distribution (mean of 1/frequency). Each of the interest packets sized ≈40kB in `ConsumerCbr`.

**ns3::ndn::Producer:** The Producer application replies to the incoming Interest packets with a data packet. The published data packet includes a name, that must match the Interest packet name. The producer application payload size is configured for 1024 bytes and each of the data packets is signed with RSA signature.

**ns3::ndn::StackHelper**: Stack Helper class is used to install ndnSIM network stack on requested nodes and configure parameters.

**ns3::ndn::GlobalRoutingHelper**: The global routing controller FIB helper interacts with the NFD FIB manager by sending special Interest commands to the manager, to add/remove a next-hop from FIB entries or add routes to the FIB. The `CalculateRoutes` class is also used to calculate the shortest path and install routes to all name prefix origins under `GlobalRoutingHelper`.

**ns3::ndn::StrategyChoiceHelper**: StrategyChoiceHelper interacts with the NFD Strategy Choice manager by sending Interest commands to the manager to specify the desired per-name prefix forwarding strategy for one, more than one or all the nodes of a topology. `BestrouteStrategy` and Multicast strategies are also used in the scenario created in this work.

**SetOldContentStore**: The default CS is configured for LRU cache policy which is an efficient data caching for popular contents. We have also applied various caching policies to mitigate the side-channel attack, such as random and probabilistic caching policies.

**ndn::L3RateTracer**: It is an NDN network-layer rate tracer to obtain the Interest and the Data packets. These metrics are obtained per packets forwarded by the node.

**ndn::CsTracer**: It is used to obtain measures of cache hits and misses on the nodes.

**ndn::AppDelayTracer**: It is used to obtain application-level delays, observed between issuing Interest and receiving Data packet.

In last, the script can be compiled by `./waf` and several simulation metrics can be also collected. To set the simulation to run multiple times and visualize its findings, a python script was also created.

### 5.2.3 ndnSIM Components

The ndnSIM 2.x is structured in three main components: *i.* ndnSIM core, where the NDN packets are managed, *ii.* the upper layer, where ndn and non-ndn applications and NetDevices (point-to-point, CSMA, wireless, *etc.*) are managed, and *iii.* NFD, which is integration to NDN protocol stack.

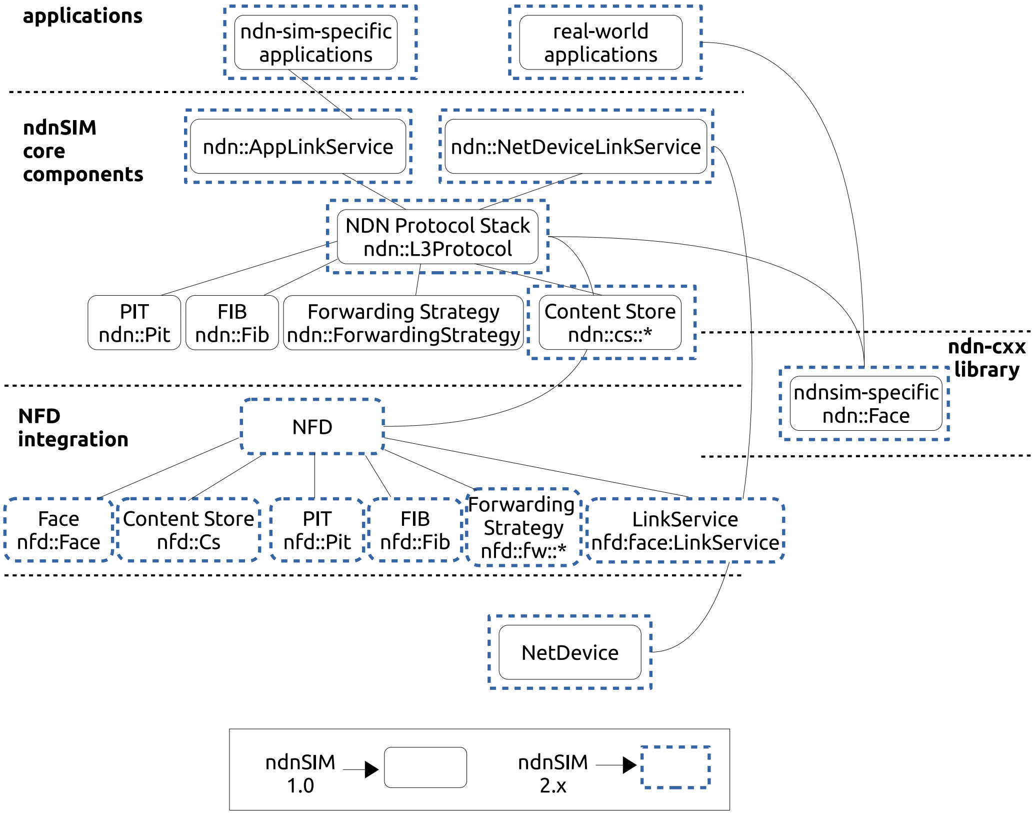


Figure 5.3: NFD integration to ndnSIM core (adapted from [10], [11] [12]).

Figure 5.3 illustrates the structural diagrams for ndnSIM 1.0 and ndnSIM 2.x. The simulator has two main releases, the ndnSIM 1.0 [10] (has no sub-release) and ndnSIM 2.x [11], [12] (has sub-releases). Besides an NFD integration, both releases have common component-level abstractions:

**ndn::L3Protocol**: An ns-3 abstraction of NDN stack implementation. It provides tracing sources from Interest and Data packets from lower and upper layers through Faces(App and NetDevice), to measure

NDN performance. The NDN protocol stack can be installed to each node in a similar way similar to IPv4 and IPv6 protocol stacks.

**ndn::AppFace**: An abstraction that implements uniform communication primitives to send Interest and received Data packets with applications.

**ndn::NetDeviceFace**: An abstraction to implement the communication with other simulated nodes such as link-level congestion mitigation modules.

**ndn::cs::***: An abstraction for in-network storage for data packets. Four main life-time tracking abstractions (::LRU, ::LFU ::FIFO, ::Random) are defined under `ndn::cs`.

**Security**: In the ndn-cxx library, the certificates and their corresponding identities are managed by `ndn::security::KeyChain`. The basic signing process is to create a KeyChain instance and supply the data packet and signing certificate name to `KeyChain::sign()` method [93].

**Applications**: ndnSIM has built-in consumer and producer applications. The applications have various parameters that can be configured by the user, used to generate user-driven network traffic.

**ndn::*Tracer**: To collect and analyze statistical information from interest, data packets, and traffic, can be done using tracer helpers.

In ndnSIM 2.x, the packet forwarding is separated into the forwarding pipelines and strategies. Through that, the NFD maintains the network for where the packets need to be forwarded by various strategies on ndn-cxx (CS, PIT, and FIB).

As a summary, the ndnSIM 2.x is supported by the following features; *i.* the forwarding and management are directly done by NFD, *ii.* the ndn-cxx library, and *iii.* the trust management with certificate packet format.

## 5.3 Named Data Network Forwarder Daemon

In ndnSIM 2.x, the NFD is a network forwarder that is used to implement the NDN protocol. The NFD extension brings modularity and extensibility to evolve the NDN applications, protocol features, and algorithms.

The main functionality of NFD is forward interest and data packets by abstracting lower-level transport mechanisms into the NDN faces. Also, the CS, PIT, and FIB can be maintained on each node by NFD. Through these components, it can support multiple forwarding strategies, face management, control, and monitor.

### 5.3.1 NFD Modules

In this section, a description of the major NFD models is presented. These modules can be upgraded by NFD releases [1].

[1] http://named-data.net/doc/NFD/current/

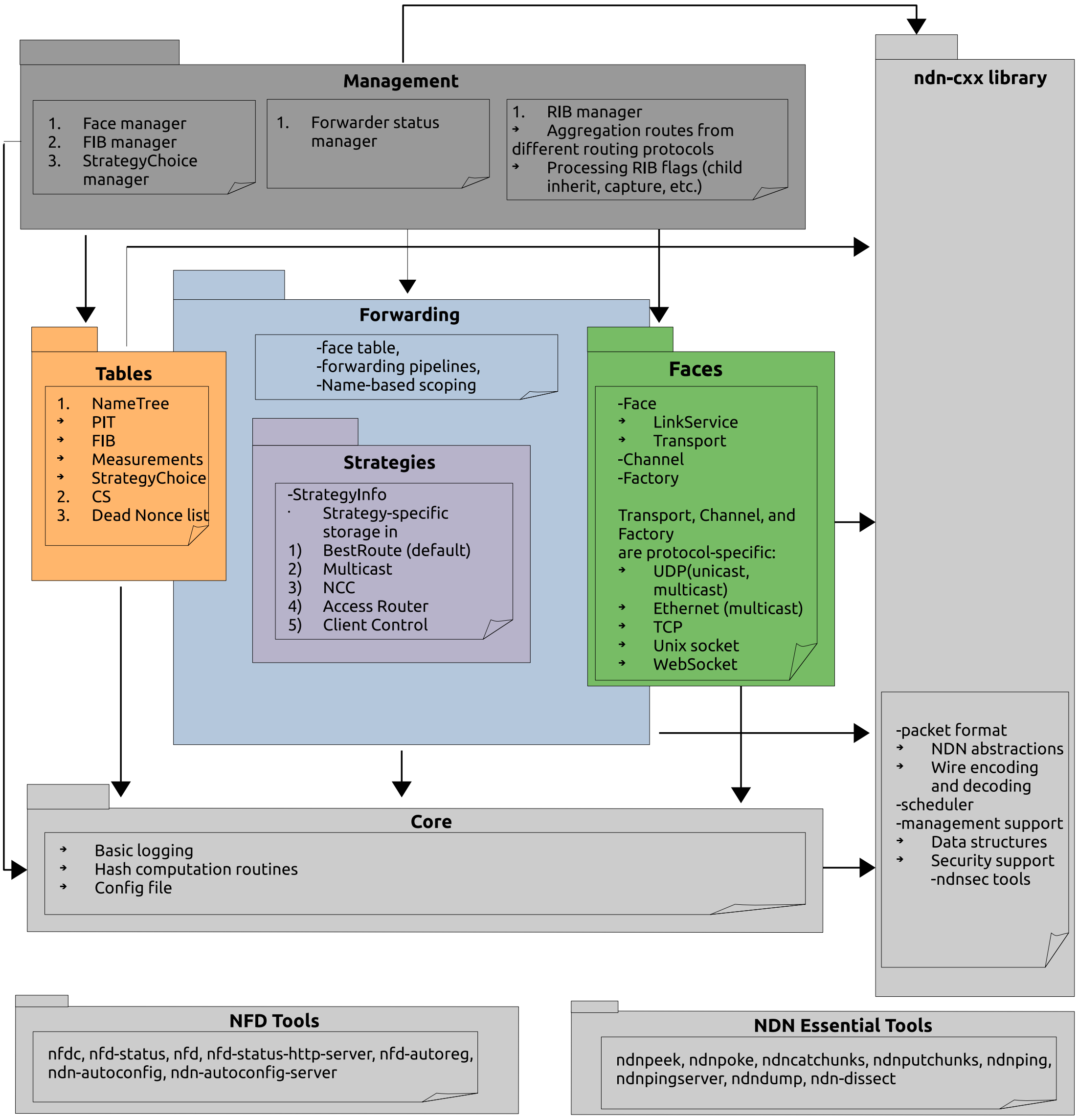


Figure 5.4: NFD modules (adapted from [13]).

Figure 5.4 illustrates the NFD modules. The NFD has five modules that implement core NDN modules (ndn-cxx, core, and tools) As illustrated in Figure 5.4, the NFD has the following inter-dependent modules:

**ndn-cxx library, Core and Tools**. The core, NFD tools, ndn-cxx, and NDN essential tools are common services shared between NFD modules. Also, these services include cryptographic hash computations, face monitoring, and DNS resolver.

**Faces**. The Face is the generalization of the interface. The packets are sent and received by each Face (similar to a physical interface). It also implements Face abstraction on top of different transport mechanisms such as *i*. An interface to communicate to the physical link, *ii*. An overlay communication channel between a remote node and NFD, and *iii*. inter-process communication between a local app. and NFD.

Through the implemented Face abstractions NDN network layer packets have a best-effort delivery service for send/receive Interest, Data, and NACK packets. Then, the Face manages underlying communication such as sockets.

The face is based on the FaceSystem-ProtocolFactory-Channel-Face hierarchy. A ProtocolFactory manages the multicast and channels (unicast) faces of the particular protocol. Channel presents a local endpoint of the protocol and owns unicast faces on a particular protocol.

**Tables**. The Table is a module that provides a data structure. The FIB is used to forward the Interest packet to the potential sources of matching Data packet. The NFD's FIB is similar to an IP's FIB except it allows for a list of the outgoing face(s) rather than a single one. Also, the outgoing faces are stored on FIB entry, which are references for forwarding.

To satisfy the next coming Interest requests, the data packet can be cached by CS. The NFD offers multiple cache replacements policies such as FIFO and LRU and resizing cache by contents.

In NFD, each PIT entry is identified by an Interest packet. The PIT entry presents a pending Interest and satisfied Interest packets. If it is not answered by a data packet, the Interest packet is considered an unsatisfied interest entry.

The Dead Nonce table is used for loop detection purposes in PIT. The stored Nonces are used to identify loop detection in PIT entries. When a Nonce is deleted (dead) from PIT entry, the Interest name and Nonce are added to Dead Nonce List.

The StrategyChoice table contains the FIB forwarding strategy selected for each namespace.

The Measurements table is used to measure PIT entries and forwarding strategies.

The NameTree is a common indexed Name for PIT, FIB, StrategyChoice, and Measurements tables.

**Forwarding**. The Face Table is a table of Faces that is managed by Forwarder (e.g. NFD). A Forwarding Pipeline operates on the Interest, Data, NACK packets, or PIT entry. It is triggered by a specific event: *i.* loop detection in PIT entries, *ii.* An Interest that needs to be forwarded out of a face, and *iii.* reception of an Interest packet. The name-based scoping is a scope control of Interest and Data according to their name-space.

**Forwarding Strategies**. A Forwarding Strategy is a decision-maker about where an Interest packet needed to be forwarded. The list of NFD forwarding strategies is defined by *i.* **best route** strategy, sends Interest packet to lowest cost upstream router(s), *ii.* **multi-cast** strategy, sends all Interest packets to all upstream router(s), *iii.* **access router** strategy, designed for aka access for local site prefix on edge routers, *iv.* **client control** strategy, a consumer application choose the outgoing face of Interest packet.

**Management**. To configure NFD and set/query, it's internal tasks are done by the management applications. NDN management also offers the capability to monitor and control the NFD. NFD management is divided by management modules, and each of them is responsible for a subsystem of NFD.

The forwarder status manager is providing statistics about the forwarder and the status of NFD such as version, NFD startup time, table entry counts, and packets (Interest, Data, and NACK) counts.

The face manager is responsible to create and destruct the faces for all protocol types. The manager can change the attributes of any face.

The FIB manager allows the authorized administer (by default its RIB management), to update/modify the FIB of NFD. The administer can ask the FIB to do: *i.* add/remove the next hop from a name prefix and *ii.* update the routing of a next-hop.

The strategy choice manager is responsible for setting and unsetting the forwarding strategies from the strategy choice table for namespaces in NFD.

**RIB Management**. The Routing Information Base (RIB) stores dynamic and static routing namespaces registration. The RIB is used to calculate the next hops for FIB entries in FIB.

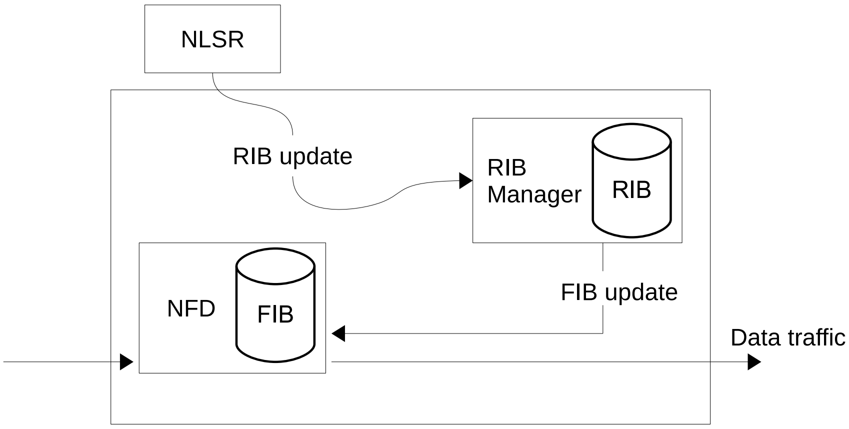


Figure 5.5: RIB manager (adapted from [13]).

Figure 5.5 illustrates, the FIB and RIB are updated by the RIB manager. The RIB can be registered by any applications, operators, and NFD. One of NDN routing protocol application called Named Data Link State routing protocol (NLSR) provides a routing protocol to populate the RIB as proposed by [40]. The NLSR calculates the routing table using link-state routing and can provide multiple faces for name prefix.

**NFD Tools**. The NFD tools are used to manipulate/update the tables, forwarding strategies, and check NFD status by pre-identified tools in NFD. New tools may be added to the NFD with updated versions for real-applications. The set of tools are used for the administrator and daemon as follows:

The `nfdc` is a tool that is used to change the RIB, FIB, and StrategyChoices table. The nfd-status is a tool used to read the NFD version and status information.

The `nfd` shows the subcommands for help, modules, and config information.

The `nds-status-http-server` is a daemon application that shows NFD status via HTTP protocol.

The `nfd-autoreg` is an application that automatically registers the name prefixes when a new needed Face is established.

The `ndn-autoconfig` is an application to run the NDN hub discovery procedure. This is a procedure that is used to detect the network changes or discover the NDN router to gain connectivity of the NDN.

The `ndn-autoconfig-server` is a server implementation for the NDN hub discovery procedure.

The ndnpeek is a simple consumer application that sends an Interest packet and expects a Data packet in return.

The ndnpoke is a producer application that creates the only payload and publishes a single Data packet.

The ndncatchunks is a consumer application that fetches Data packets with segments.

The ndnputchunks is a producer application that creates Data packets with segments.

The ndnping and ndnpingserver are testing tools for the reachability of two nodes.

The ndndump is a network traffic analyzer tool that captures NDN packets.

The ndn-dissect is a packet for the inspector of NDN.

### 5.3.2 Content Store

The content store is used to cache the data in ndnSIM. The requested data packets are placed in the cache as long as according to the configured cache policy.

The CS is implemented in nfd::cs::CS class. This has two main sub-classes as followings: *i.* lookup table and *ii.*replacement policy.

**look-up table**. In CS, the lookup table is a name-based index of CS entities.

**replacement policy**. The cache replacement policy manages the limits of CS capacity. When it is full of cached contents, it discards (flushes) the cached contents to able to cache new entities. Also, the NFD offers multiple cache replacement policies by ndnSIM 2.x, including LRU (least recently used) and priority-FIFO (first-in-first-out) policies.

The LRU cache policy flushes the least recently used contents from the CS. Also, LRU uses one queue to obtain data usage in CS. Its table iterator is relocated to the tail of the queue when an entity is used or refreshed. When an entry needs to be evicted, its table iterator and table entry are erased from its queue.

In priority-FIFO, three queues are used to track the data packets in CS. These queues are identified by the following *i.* unsolicited: a queue that contains the entries with unsolicited contents, *ii.* stale: a queue that contains entries with stale contents, and *iii.* FIFO, a queue that contains entries with fresh content. If an entry matches one of these queues, it only appears once in that queue.

The table iterator can be stored in any of the queues. This establishes a relationship between the table and the queues. The operations are classified by the followings:

- When an entry is inserted in the table.
- When an entry evicted and its table iterator erased from the head of queue and entry also erased from the table.
- When an entry became stale, the table iterator is moved from the FIFO queue to the stale queue.

- When an unsolicited entry is updated with solicited content, the table iterator is moved from unsolicited queue to FIFO queue.

The CS also offers a probabilistically caching contents of the nodes. This can be adapted to LRU, FIFO, and other policies. These cache policies are selected accordingly to the application's configuration.

### 5.3.3 NDN Testbed

The NDN-testbed is created to test the NDN paradigm and daemons underlaying of TCP/IP. Currently, the NDN-testbed consists of 42 NDN represented routers on the research institutions and the universities [14].

Also, the NDN-testbed has been established and maintained at the University of Minho since 2016. The main reason to join the NDN-testbed was to develop this work scenario to run over the NDN-testbed. However, the scenario implementation faced some challenges considering the NDN-testbed updates and getting permission from other nodes.

Therefore, the extracted NDN-testbed topology was used to simulate the applications over ndnSIM instead of the NDN-testbed.

## 5.4 Implementation

To achieve the work objectives, some components were built in *C++* 11 and in *Python* in order to integrate the *ndnSIM* simulation framework. The components are organized according to the type of component:

- **NDN Applications** - includes a *BadGuy* (the attacker application) able to perform brute force random attacks to a pre-configured set of name prefixes, *vondn-app* (VoNDN Caller/Callee) and *ndntube-app* (a stream consumer application)

- **Core Components** - includes a new content-store implementation, called *content-store-privacy*, that implements the DaD algorithm proposed in this work

- **Scenario programs** - a simulation main program, for each simulated scenario, including a *vondn-simulation* and a *ndntube-simulation*, that loads the proper network topologies and creates and configures the simulation respectively for VoNDN and NDNtube application scenarios

- **Orchestration scripts** - ex: `run.py`, written in *python*, this script is used to repeat each simulation scenario for a predefined number of times, for each different set of parameters, being able to change the topology, the cache policies, application instances, simulation duration, etc. The *python* scripts also process the simulation results and build the expected graphs.

Following the ndnSIM best practices, all components were kept on external ndnSIM ”scenario” directory, to isolate them for future ndnSIM version upgrades. The components are organized in subdirectories according to the type of component as describe in Table 5.4.

| **ndnSIM `ndnSIM/scenario` directory** | |
|---|---|
| **sub-dir** | **Components** |
| `./scenarios` | Contains the scenario main programs:<br>**`vondn-scenario.cpp`** and **`ndntube-scenario.cpp`**.<br>All created simulation programs placed in this directory will be compiled by **`./waf`** command. Contains also a sub-directory called “*disabled*” used to place the disabled scenarios not to be compiled. |
| `./extensions` | Contains the application programs and NDN core components:<br>**BadGuy.{hpp,cpp}**, **vondn-app.{hpp,cpp}**, **ndntube-producer.{hpp,cpp}** and **ndntube-consumer.{hpp,cpp}**.<br>The applications created and the NDN core components are placed here. The extensions are also compiled by *./waf* command and linked to scenario main programs. |
| `./topology` | Contains all network topologies used in the simulations:<br>**simple.txt**, **small-tree.txt**, **tree.txt**, **testbed.txt**, **vondn.txt** and **streaming.txt**.<br>Some topologies, like simple and small-tree are useful to run small tests for debugging or visualization. Others are more realistic, like NDN-testbed, that was derived from NDN real testbed as already explained. |
| `./build` | Contains generated executable files for each simulation:<br>**`vondn-scenario`** and **`ndntube-scenario`**.<br>When the scenario and extensions are successfully compiled by `./waf`, an executable file is generated for each scenario’s main program. |
| `./results` | Contains results files built by the simulations, during each run.<br>Results are generated by *tracers* objects, carefully created and defined in the simulation main program. Results are stored as compressed files (.bz2) |
| `./graphs` | Contains a set of R scripts used to collect and process the result files generated by the simulation: **preprocess.R**, **rates.R**, **cachehitratio.R**, ...<br>The R scripts produce graphs in several formats (*ex.*: .pdf and .eps) inside this folder. |

Table 5.4: Implementation: major components organized by type and folder.

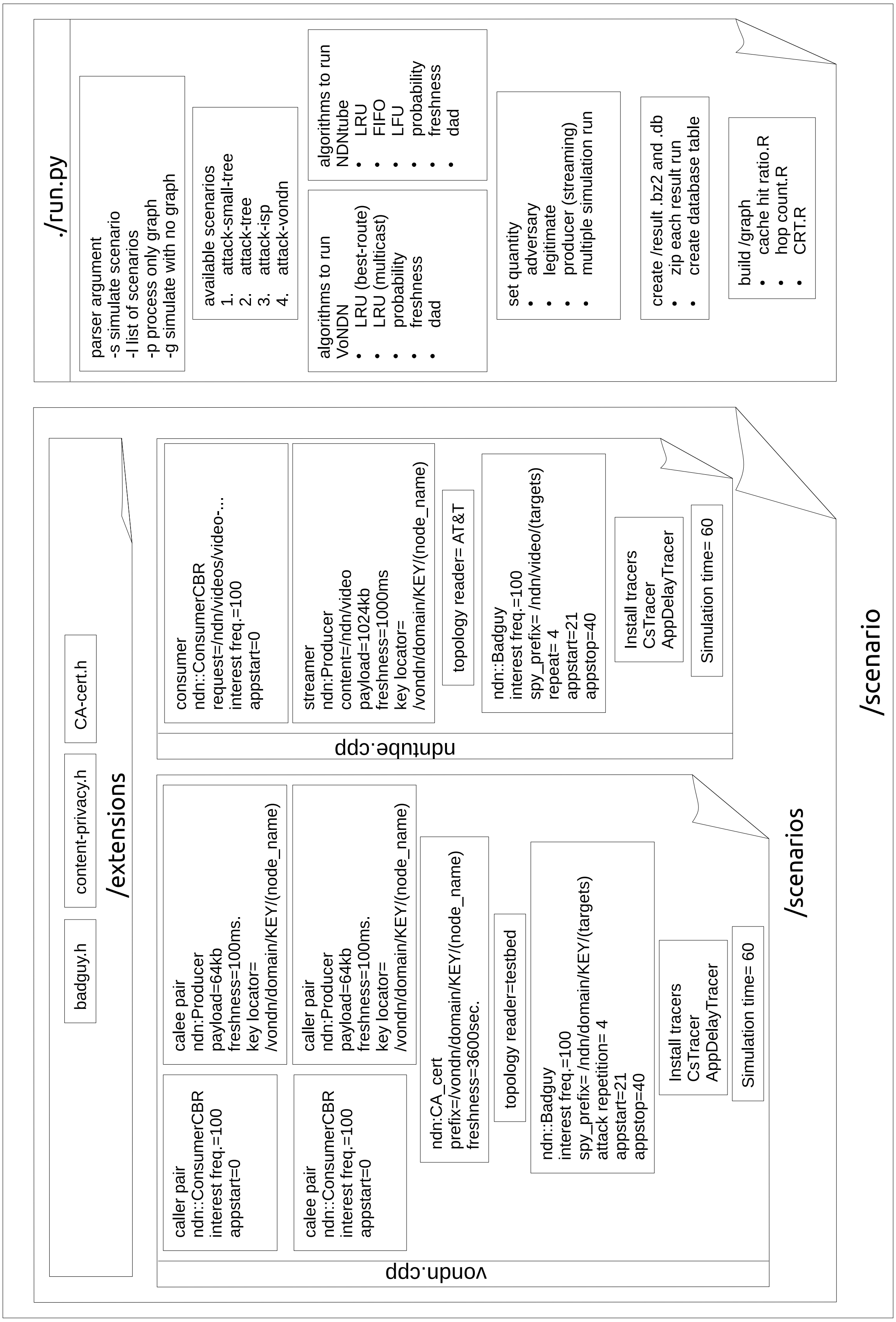


Figure 5.6: Scenario implementations.

Figure 5.6 illustrates the implementations of the `/scenario` directory for this work. In this implementation, the VoNDN and NDNtube applications were placed in `/scenarios` and their extensions were placed in `/extensions` directory. The `./run.py` was configured to set several simulations for different available scenarios, algorithms. Also, a python script was used to create database files (`.db`) to be analyzed by R scripts (cache hit ratio, hop counts, and content retrieval time) and graphs (`.eps`).

### 5.4.1 NDN Applications

Figure 5.7 shows the C++ classes used. The top parent class is `ns3::ndn::App`, defines common methods to all applications, like *startApplication* and *stopApplication*, for instance. Also, some virtual methods that have to be defined later on all subclasses, like *onInterest* or *onData* that are called when the interest packet or data packet is received by the application. The classes `ns3::ndn::Producer` and `ns3::ndn::Consumer` inherit from the it, implementing the basic functionality of a producer and a consumer. A more specific consumer application, already present in ndnSIM, is the `ns3::ndn::ConsumerCbr`, that request interest with a configured fixed frequency, originating a constant bit rate stream of data packets to be received. This application is therefore used as a base class for NDNtube consumers but also for the `BadGuy`.

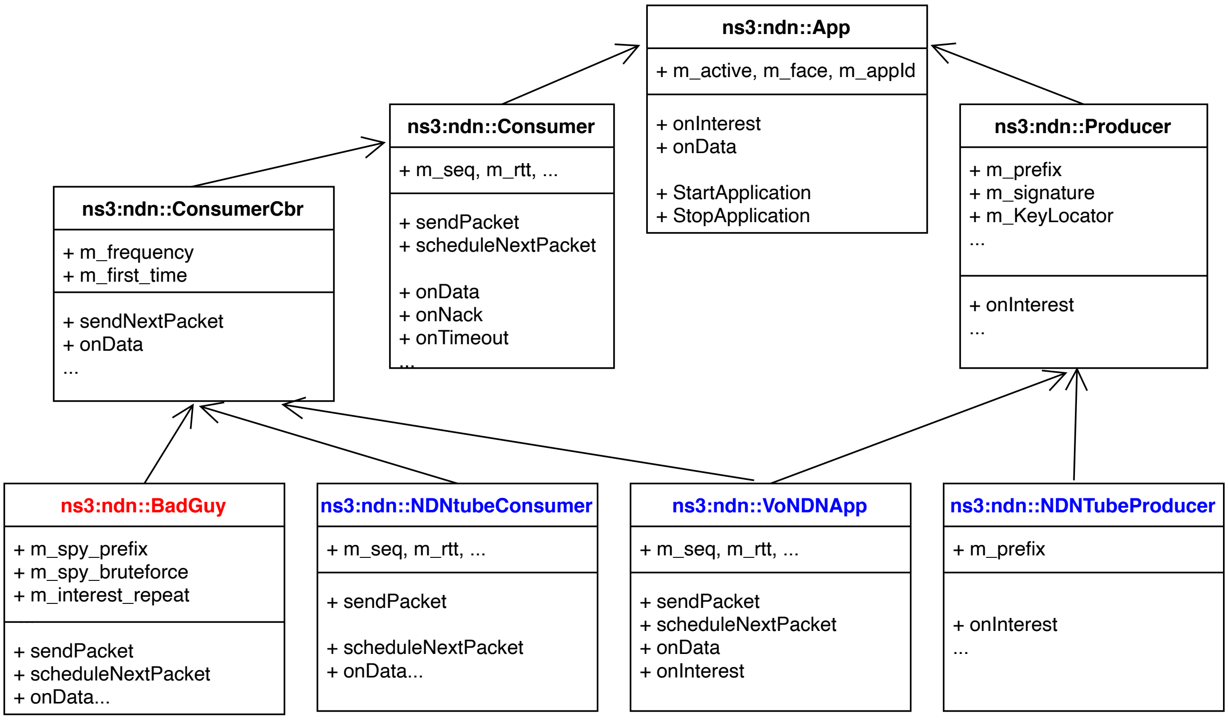


Figure 5.7: Applications: class diagrams.

The class `ns3::ndn:BadGuy` was created to implement the brute force algorithm already described in the previous Chapter 4. Before starting some attributes may be configured to change the behavior of the application, like the prefix to be spied on (*SpyPrefix*), the brute force behavior activation (*BruteForce*), and the number of repetitions for each interest (*InterestRepeat*). A number of valid names are also, to be added to the prefix. If not specified all possible node names are used together with the spy prefix to create potential target names to attack. When the application starts a target name is selected randomly, and an interest packet is issued, for a number of repetitions. Since BadGuy inherits from ConsumerCbr, the rate of requests is defined by the `m_frequency` configured value.

Regarding `ns3::ndn::NDNtubeConsumer` it simply extends the behavior of the existing ConsumerCbr, to simulate a client requesting a video stream. Each stream is identified with a prefix name, and each packet on sequence is numbered. The NDNtube consumer application increments the sequence number and sends a new Interest packet for that sequence number. The process repeats at each time interval. The `ns3::ndn::NDNtubeProducer` is also a simple extension of `ns3::ndn::Producer`. The producer producing the video with */ndntube/videos/(video-..)* to the NDNtube consumers.

Finally, VoNDNApp is also based on the same base classes. The first version was only a combination of the ConsumerCbr and the Producer on a single node, simultaneously sending Interest packets and producing data packets for interests received. The caller node, running an instance of the VoNDNApp, combines the name of the callee and the configured call-id into a full name of the form */vondn/<callee-name>/call-id/<call-id>* to identify the voice data packets originated by the callee and continuously send interest for that name with a sequence number as a suffix. In a similar way, the callee requests interest on data packets named */vondn/<caller-name>/call-id/<call-id>*. This simulates the media streams for voice/video conversation. Later the signaling was added when the call starts. The caller sends a special interest packet with call-request (*/vondn/<callee-name>/call-id/<call-id>/call-request* the callee answers back with a data packet accepting the connection. Before starting requesting media data packets, they both request each other public key certificate using the name */ndn/domain/vondn/KEY/<node-name>*. The adversary may, therefore, try to target the certificates on cache but also the media data packets of the conversation. A voice conversation may have periods of silence alternated with periods of talk, while a uniform distribution of interest request is more adequate to simulate voice/video calls. The default behavior used is a uniform distribution.

### 5.4.2 Core Components

For the implementation of the DaD algorithm, the first question that must be addressed is where to add it, which components in the simulator to modify, or new components to add to implement the required functionality.

Considering the ndnSIM software architecture described in Figure 5.7 and 5.8, we can see that there is a set of major components that implement the core elements of an NDN router, but also a set of pluggable components that can be added without interfering with major blocks. Core elements are, for instance, the data structures that support all the required tables (CS, FIB, PIT), the Face objects that abstract the connectivity to external routers and internal components (applications, tables), a major pipeline for processing interest packets and another one for processing data packets. The pluggable components allow inserting new cache policies and new forwarding strategies, without modifying the internal structure.

Since the DaD algorithm changes the way information is cached or not in CS or the way the information is retrieved and delivered from CS when under attack, it seemed natural from the beginning to implement it as a new content store policy. The approach followed was therefore to design a new class `ns3::ndn::cs::Privacy` (*content-store-privacy.hpp*, *content-store-privacy.cpp*), that in practice wraps the expected behavior of existing content stores, including the set of cache replacement policies already available, and adds in some extra functionality. When interest arrives a name lookup occurs, it is an opportunity to do a real lookup on supporting content store implementation and run the DaD algorithm. The first task is to update the cache hit and misses measures and the cache hit ratio for the incoming face. Next, a decision is taken regarding attack detection. If the state is considered to be no an attack is detected, the data packet is returned normally. If an attack is detected, the action is taken according to its severity as previously described.

### 5.4.3 Scenario programs

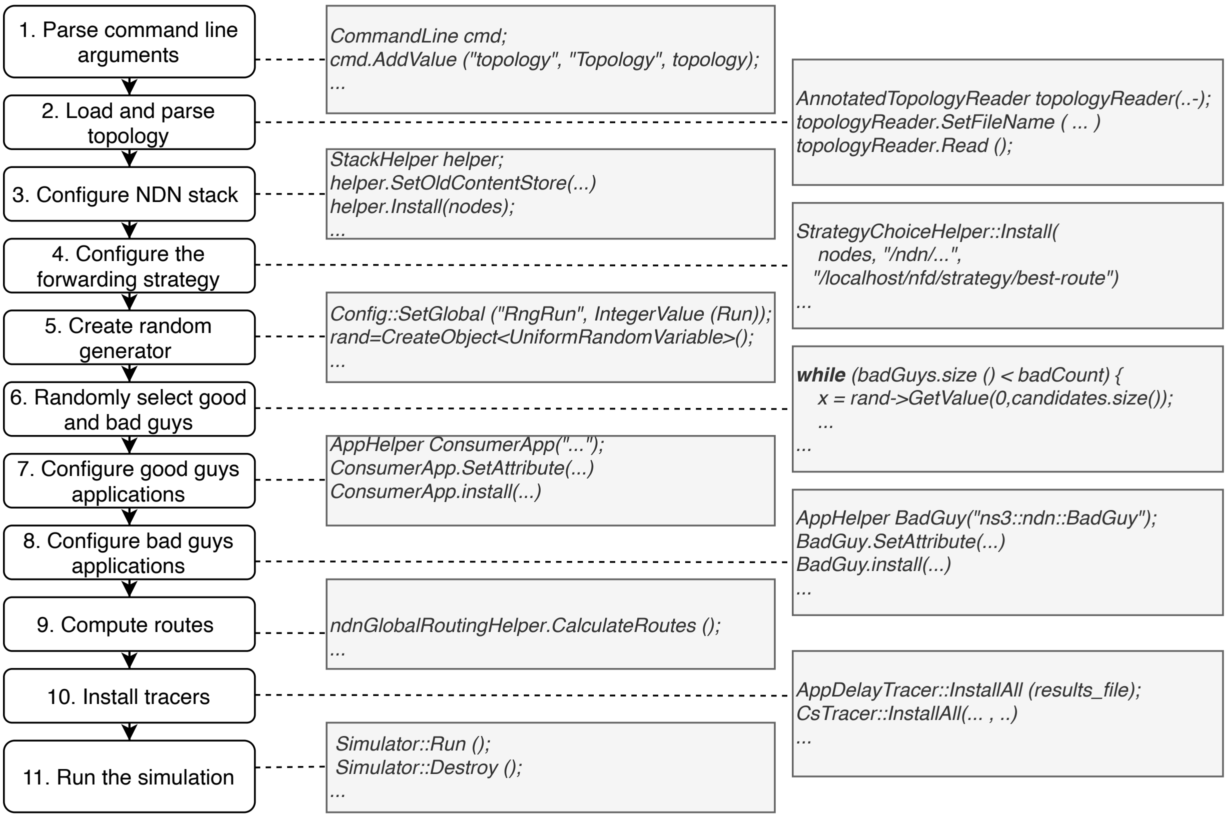
The scenario programs are also written in C++ and they follow the usual structure of a ndnSIM simulation, also similar to normal NS-3 simulation. The process is illustrated in Figure 5.8 in a generic way. The common part of the structure can be detailed by the following sequence of actions:

1. **Parse command-line arguments.**

   The following are examples of the considered parameters:

   - *topology* - the name of the topology file to use;
   - *algorithm* - mitigation algorithm to use (if any);
   - *badCount* - number of bad nodes, running the adversary application, to consider in the simulation;
   - *prodCount* - number of regular nodes, running normal application to consider for producers;
   - *goodCount* - number of regular nodes, running normal application to consider for consumers;
   - *folder* - folder to save the simulation result files;
   - *run* - the number of the simulation run.

2. **Load a topology from an external file.**


1. Parse command line arguments
2. Load and parse topology
3. Configure NDN stack
4. Configure the forwarding strategy
5. Create random generator
6. Randomly select good and bad guys
7. Configure good guys applications
8. Configure bad guys applications
9. Compute routes
10. Install tracers
11. Run the simulation

```
CommandLine cmd;
cmd.AddValue ("topology", "Topology", topology);
...
```

```
AnnotatedTopologyReader topologyReader(..-);
topologyReader.SetFileName ( ... )
topologyReader.Read ();
```

```
StackHelper helper;
helper.SetOldContentStore(...)
helper.Install(nodes);
...
```

```
StrategyChoiceHelper::Install(
   nodes, "/ndn/...",
   "/localhost/nfd/strategy/best-route")
...
```

```
Config::SetGlobal ("RngRun", IntegerValue (Run));
rand=CreateObject<UniformRandomVariable>();
...
```

```
while (badGuys.size () < badCount) {
   x = rand->GetValue(0,candidates.size());
   ...
...
```

```
AppHelper ConsumerApp("...");
ConsumerApp.SetAttribute(...)
ConsumerApp.install(...)
```

```
AppHelper BadGuy("ns3::ndn::BadGuy");
BadGuy.SetAttribute(...)
BadGuy.install(...)
...
```

```
ndnGlobalRoutingHelper.CalculateRoutes ();
...
```

```
AppDelayTracer::InstallAll (results_file);
CsTracer::InstallAll(... , ..)
...
```

```
Simulator::Run ();
Simulator::Destroy ();
...
```

Figure 5.8: Scenario programs: common structure.

Topology is loaded using and object of class `ns3::AnnotatedTopologyReader`. The reader validates the topology and loads it to memory as a set of nodes and links with properties.

3. **Configure NDN stack and install it on each node of the topology.**

   The helper class `ns3::ndn::StackHelper` is used to configure NDN stack. One of the things that must be configured is the cache policy to use on nodes. The ndnSIM simulator already includes a set of cache replacement algorithms, like LRU (Least Recently Used), FIFO (First In First Out), LFU (Least Frequently Used), Random, etc. One of the available policies may be selected and applied. For instance, considering only as an example the LRU, the `ns3::ndn::cs::Lru`, `ns3::ndn::cs::Probability::Lru` and `ns3::ndn::cs::Privacy::Lru` classes can be used. This last one implements also the DaD algorithm.

   The cache policies are applied to all nodes, but the bad nodes (adversaries) need always to change it to a *no-cache* policy because they must avoid their cache during the timing attacks on target names.

4. **Define and apply forwarding strategy to all nodes.**

   Only two strategies were considered to be applied on the nodes, using the helper class `ns3::ndn::StrategyChoiceHelper`: the best route strategy (named

*/localhost/nfd/strategy/best-route*) and the multicast strategy (*/localhost/nfd/strategy/multicast*). This strategy is used to populate the FIB with the name prefixes produced by the producer nodes.

5. **Configure the random generator seed, based on the simulation run, and create an object of class `ns3::UniformRandomVariable` for random number generation.**

   Following the NS-3 design, the random seed for (pseudo) random numbers is fixed for each run and depends only on the *run* number, making it reproducible. This is a very important design feature of NS-3. Different runs use different seeds and therefore generate different sequences of random numbers.

6. **Randomly select**.

   *prodCount* number of nodes for producers, *badCount* number of nodes for bad guys, *goodCount* number of nodes for consumers.

7. **Configure and install applications on bad guys nodes**.

   Usually, a special `ns3::ndn::BadGuy` consumer application that implements the brute-force attack algorithm. A start event and a stop event must be scheduled to start the application at the beginning of the attack and stop it at the end of the attack period. The attack period is predefined for each scenario.

8. **Configure and install applications on good nodes, both producer and consumer**.

   Here the configuration may change according to each application and will be explained later on for each application scenario in the following sections. Again, start and stop events must be generated to allow the applications to start and stop as expected. Applications start a small random amount of time after the simulation starts and end with simulation.

9. **Compute routes and populate FIB on all nodes**.

   Using a `ns3::ndn::GlobalRoutingHelper` object, the routing can be configured. All names produced by good guys applications are registered and short path trees are calculated using method *CalculateRoutes*. Routes are installed for all prefixes origins.

10. **Install Tracer objects to collect and dump results to the result files.**

    Examples of the tracers used are `ns3::ndn::AppDelayTracer`, that can collect and register application-level delays, and `ns3::ndn::CsTracer`, that can collect and register information about cache events, like cache hits and cache misses. Results are stored on a zipped file, in the results folder, named using a combined string containing the topology name, number of good and bad guys, and simulation run for easy post-processing.

11. **Define simulation time and run the simulation**.

    Schedule a stop event to end the simulation, and start it by calling `Simulator::Run()`.

Based on this common structure, specific scenarios were produced. The two main scenario programs created are including a `ndntube-scenario` and a `vondn-scenario`.

**ndntube scenario**. The `ndntube-scenario` program creates only one producer and a set of *goodCount* consumers.

The NDNtube producer is installed on the previously randomly selected producer node. It is configured to produce a stream of video with prefix name *"/ndntube/videos/..."*. Packets are signed by the producer, and by default have a freshness of $1000\text{ms}$ and a virtual payload size of $1024$.

An NDNtube consumer app is also installed on all randomly selected consumer nodes. They are all configured to consume a specific stream produced by the NDNtube producer.

**vondn scenario**. For the `vondn-scenario` program, nodes engage in a conversation on pairs. On each simulation a total of *goodCount* nodes is engaged on *vondn* conversations. Each pair of nodes is randomly chosen from the set of eligible consumer ones. For each pair, first, the callee is selected, a *call-id* is generated, unique for each call, and then the caller is also selected. Each node signs the data packets using a key named */ndn/domain/vondn/KEY/<node-name>*. Each node is simultaneously a producer and a consumer inside the call. Packets produced by a node engaged in a VoNDN call use a naming format */vondn/<node-name>/call-id/<call-id>*.

A Certification Authority, named *CACert*, is also placed randomly on a node in the topology. The *CACert* provides certificates for all keys with prefix *"/ndn/domain/..."*. Each node that needs to validate a key certificate, issues an interest with the key name. If the certificate data packet is not cached on any node, the interest will reach the *CACert* node that answers it with a signed data packet. For simplicity, only one *CACert* node is used, instead of a full hierarchy of public key certification authorities (CA).

### 5.4.4 Orchestration scripts

As mentioned, a Python script called `run.py` is used to run a specific simulation scenario a specific number of times for each configuration. The script can run the same program for different caching policies, defined in the script and including the DaD algorithm, for different topologies, for a different amount of good and bad guys, and for a number of runs. For each simulation, one or more result files are produced, by each tracer.

After running all simulations, the `run.py` also initiates the post-processing of all results. The first step of result post-processing consists in transforming the zipped text files into a structured database table. Each result file is unzipped and loaded to an `SQLite` .db file. Each log text line is separated in a set of fields, into a record inserted in a table, for further processing.

The next phase of post-processing is done using the R script and also called from the `run.py` script. Data on tables are summarized and aggregated by categories. For instance, delay values or cache events can be aggregated by node type, for good guys or for bad guys, and statistics like average and others can be computed for all the simulations runs. Intermediary results are stored again on the disk, now as R data files.

The last phase of post-processing is to produce the graphs. Again this is done by specific R scripts for each graph, called by `run.py`, using the data files stored in the previous phase.

**Source code**. This work was implemented using the external scenario directory of ndnSIM. Through that, only the implementations were complied with instead of compiling complete ndnSIM. Also, the scenario implementation support is an open-access for future work implementations. The scripts C++, Python (2.7.12), SQL, and R (3.5.2) can be accessible at the author GitHub account — https://git.io/Jey7B.

## 5.5 Summary

To create the scenario strategies, the ndnSIM framework and this work's main scenario instruments are illustrated in this chapter. An overview of the simulation framework and NFD components were presented, together with the modules created to implement.

Next, the implemented scenario findings and results will be presented based on the NDNtube and the VoNDN applications.

# 6

# SCENARIOS AND RESULTS

In this chapter, a set of experimental results obtained using simulation are presented and analyzed. The chapter starts by defining the experimental objectives. Then, the scenarios that were developed for testing, with the NDNtube and VoNDN applications, attack models, and countermeasures, are detailed. Also, in these scenarios, network topologies (AT&T and NDN-testbed) are used to collect and analyze more realistic results.

To mitigate the attack in both applications, the traditional and DaD countermeasures are applied and their results are compared in this chapter.

## 6.1 Context

To evaluate the attack and countermeasures the experimental scenario objectives are identified. The experiment objectives are directly related to the work objectives and are identified as the followings:

1. Reproduction of the applications to be presented as NDNtube and VoNDN.

2. Development of an attack application that is utilized by brute-force and randomized probing operations.

3. Implementation of the attack scenario on NDNtube and VoNDN applications to analyze attack findings.

4. Simulation of the scenarios using real data-set topologies such as AT&T and NDN-testbed.

5. Implementation of the DaD algorithm and compare the results with statically configured countermeasure ones (*e.g.* probabilistic caching) to mitigate the brute-force attack.

**Scenario configurations**. In implementations, different cache policies and forwarding strategies are used to analyze/understand the attack findings and adversary behavior on NDNtube and VoNDN.

i. In NDNtube, the attack scenario is analyzed on different NDN content store policies (LRU, LFU, and FIFO), respectively. In each policy, the performance of attack and adversary face detection results (CRT, CHR, and hop count) are analyzed for the NDNtube application. To mitigate the attack, statically countermeasures (`nfd:probabilistic` and `nfd:freshness`) methods are compared with the DaD (`nfd:dad`) for the distribution and mitigation performance.

ii. In VoNDN, the attack performance, and adversary behavior are analyzed under the following NDN forwarding strategies: *a*. Best route strategy: The best route strategy forwards an Interest packet to the upstream with the lowest routing cost. *b*. Multicast strategy: The NDN multicast strategy forwards every Interest to all upstreams (next-hops), indicated by the supplied FIB entry.

    The adversary face detection methods (CRT, CHR, and hop count) are analyzed in VoNDN. Also, the statically configured countermeasure (`nfd:probabilistic`) is compared with the DaD (`nfd:dad`) in regard to mitigation and distribution performance.

## 6.2 Scenario Implementations

The cache policy and the scope of the attack can be different for each application. Therefore, in this work, the NDNtube-like and VoNDN-like applications were developed to analyze adversary node behaviors during the attack. To generate network traffic, the applications were simulated using real data-set topologies. Through these implementations, the adversary detections were suggested for streaming and VoNDN applications respectively.

The NDNtube and VoNDN applications were reproduced with their basic features to obtain the attack findings. To understand and analyze an adversary behavior, the consumer nodes are sending their packets within flow with a packet rate of 100 packets/s in NDNtube and VoNDN testing scenarios. This means a constant bit rate of 819.2 Kbps per flow for a packet size of 1024 bytes, and 51.2 Kbps per flow for a packet size of 64 bytes. In these scenarios, the consumers are constantly consuming content which creates a negligible gap between sending and receiving the packets.

**NDNtube-like attack scenario**. Figure 6.1 illustrates the NDNtube-like streaming application in perspective on how the design of applications and adversaries can be done. In this scenario, a single producer, producing content to consumers, mimics streaming video-like applications. Also, the consumers are configured with a unique streamer id and request the contents with *streaming_app/videos/video-1/unique-streamer-id* from the producer. In the NDNtube application, each of the gateway routers is used to cache the video segments that can be retrieved by the consumers.

In the streaming scenario demonstration (Figure 6.1), the adversary pursues the video segments cached by gateway routers previously. Therefore, the adversary (*adversary-1*) can knowledge about the popularity of streamed contents, recently cached by the edge routers these requested by streamers (*streamers-1*). In the

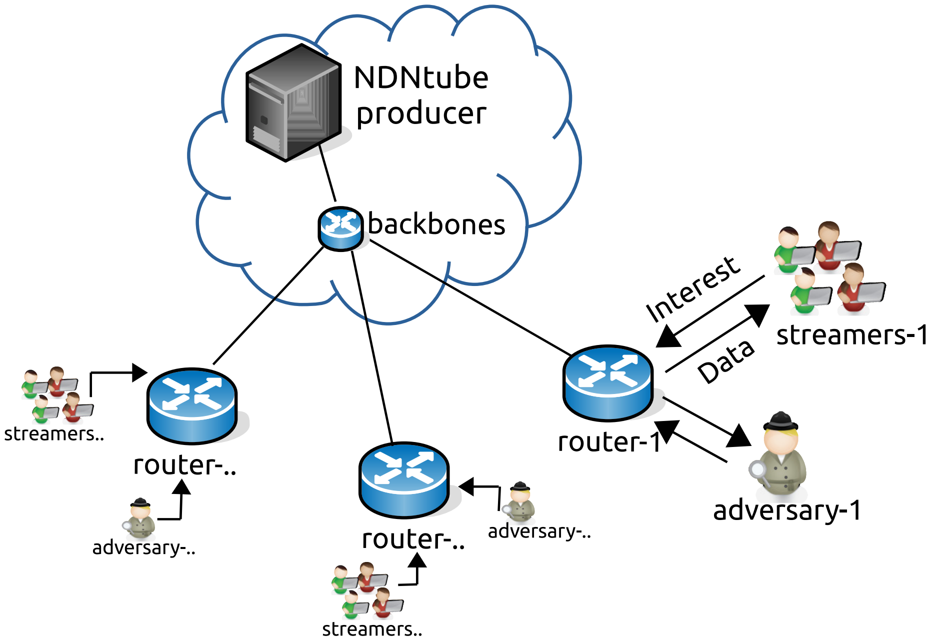


Figure 6.1: NDNtube-like application attack design.

attack, the adversary only probes the target (*e.g. streaming_app/videos/video-1*) then the gateway routers (edge, neighbor, and away) replies a video segment (*e.g. streaming_app/videos/video-1/%00%12%34*) that has been cached recently. In an ideal attack, the adversary repeats each target at least four times to differentiate that the target has been cached by the first router or the away routers. Also, other adversaries are (*adversary-...*) attacking other streamers (streamers-..) to conclude that the streamed contents are cached by the edge, neighbor, and away routers. In this attack, the adversaries only determine the popularity of the streamed contents by their locations.

Table 6.1 shows the configurations of the attack on the NDNtube-like application on AT&T topology. The AT&T topology has 625 nodes in total, from which 156 leaves (consumers), 140 evils (adversaries), and one producer (NDNtube producer) were selected randomly for each one of the 10 simulation runs. The attack was applied to default CS policies (LRU, LFU, and FIFO) to evaluate the behavior of the attack (in total 30 runs). The adversaries have as targeted the video segments named as */ndntube/videos/video-...* and cached by gateway (edge) routers. In the NDNtube-like attack scenario, the adversaries, the streamers, and the producer were randomly selected for each scenario run, the results collected from each of CS policy (LRU, LFU, and FIFO) and 30 simulation runs in total. The targeted video segments were retrieved by brute-force and each target request was repeated 4 times to increase the attack success. When the attack is finished, the adversaries compare the CRTs these collected from the gateway routers to decide where/when the certificates have been cached.

**NDNtube configuration**. In the NDNtube application, each of the streamed content (*e.g.* pre-recorded video and live-video) is produced by a single content producer. Each of the content is signed by the pro-

Table 6.1: NDNtube attack scenario configuration.

| | |
|---|---|
| **Network topology** | AT&T |
| **Total nodes** | 625 |
| **Backbones** | 221 |
| **Attacked edge router** | 108 routers |
| **Target quantity** | ≈55% of total consumer nodes |
| **Adversary quantity** | ≈45% of total consumer nodes |
| **Streaming producer** | /ndntube/videos/ |
| **Consumers** | /ndntube/videos/... |
| **Targets** | /ndntube/videos/... |
| **Attack repetition** | 4 for each target |
| **CS policies** | * LRU<br>* LFU<br>* FIFO |
| **CS size** | 1000 packets |
| **CRT decisions** | * cached by edge router<br>* cached by neighbor router<br>* cached by away router |

ducer but it is not validated by a certificate authority. The consumers are sending their packets at a constant rate (100 packets/s) to the content producer. The producer virtual payload size is defined as 1024 bytes for streamed contents. Note that, because NDNtube is implemented only for the attack purposes, encoding/decoding audio and video compression formats (*e.g.* H.264, MPEG-2) were not implemented.

The attack success and detection may be different on NDN CS policies. In NDNtube, the attack scenario was implemented using ndnSIM content policies: *i.* LRU, which removes the least recently used streamed content segment when the CS is full. *ii.* LFU, in which the least frequently used cache block is removed whenever CS is overflowed. *iii.* FIFO, where streamed contents are evicted in the same order as they come into CS.

**VoNDN-like attack scenario**. Figure 6.2 illustrates the two-way communication VoNDN application in perspective on how the design of applications and adversaries can be done. In this scenario, the callee and caller publish their interest and data packets to each other without relying on any middle transmission server such as a SIP proxy or a SIP server. To establish a call, the callee and caller exchange data using each-other their unique-call-id (*/vondn/user/unique-call-id*). To authenticate the conversation, the CA publishes certificates (*e.g. /ndn/domain/vondn/KEY*) and these can be cached by gateway routers. Also, the callee or the caller can authenticate themselves by digital signing data packets, using public-key cryptography. For that public-key certificates are required, issued by CA authority.

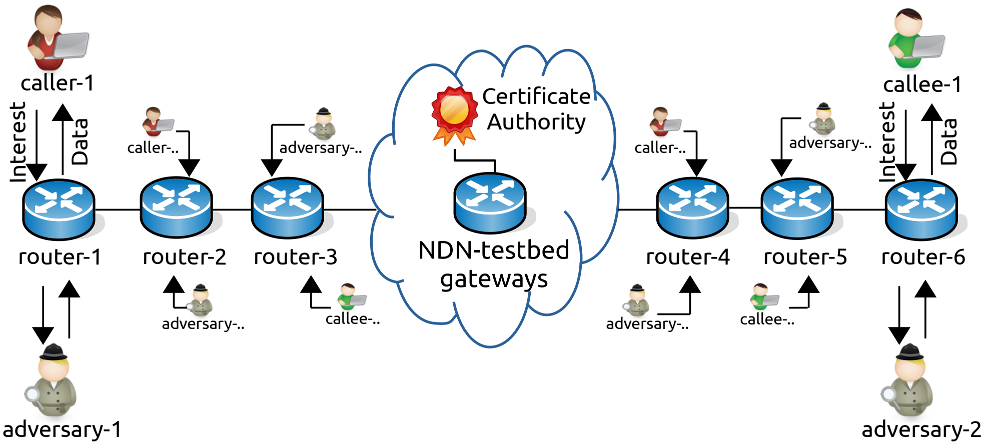


Figure 6.2: Trusted VoNDN-like application attack design.

In a trusted-VoNDN application demonstration (Figure 6.2), the adversary nodes (*adversary-1* and *adversary-2*) pursues the targets that are recently cached by gateway routers to knowledge Callee (*e.g.* Bob) and Caller (*e.g.* Alice) locations. Each of targeted certificate (*e.g. /ndn/domain/vondn/KEY/[cert-name+digest]*), was requested and delivered to the caller or callee. Caller (*caller-1*) public-key certificate is used by the Callee (*callee-1*), while the callee-1 public-key certificate is used by the caller-1, to validate the signature and authenticate each-other packets. In this attack, the adversary probes multiple certificates (targets) by brute-force and randomly to maximize the success of the attack as it was previously designed and presented in Chapter 4. Additionally, the attack is repeated at least four times to differentiate the target that has been cached by the edge router or the other routers (neighbor and away). Finally, the adversary can analyze these four CRT samples to conclude that the target has been cached or not by the edge router. Also, other adversaries are (*adversary-...*) attacking to other caller and callee (*callee-...* and *caller-...*) to conclude that their certificates are cached by the edge, neighbor, and away routers. For instance, the adversary-1 identifies the following locations: *i.* If the Callee-1 certificate is retrieved from router-1 the target is located at an edge router, *ii.* If the Callee-1 certificate is retrieved from router-2 the target is located at a neighbor router, and *iii* If the Callee-1 certificate is retrieved from router-3 the target is located at away router(s). These decisions are also taken by adversary-2 for router-4, 5, and 6.

Table 6.2 shows the adversary configuration for the attack scenario. In this model, the selected adversary nodes ($\approx$40%) targeted the certificates (*e.g. /ndn/domain/vondn/KEY/[cert-name+digest]*) that were previously cached by routers and produced by certificate authority (*e.g. /ndn/domain/vondn/KEY*). Each named certificate has an unique sha-256 cryptographic digest function (*e.g. .../sha256digest=fde78cbdff...c4106*) to authenticate the callee and the caller's identity.

Table 6.2: VoNDN attack scenario configuration.

| **Network topology** | NDN-testbed |
|---|---|
| **Total nodes** | 462 |
| **Attacked edge routers** | 42 routers |
| **Legitimate nodes** | ≈60% of total consumer nodes |
| **Adversary nodes** | ≈40% of total consumer nodes |
| **Certificate authority (CA)** | /ndn/domain/KEY/ |
| **Targets** | /ndn/domain/vondn/KEY/... |
| **Attack repetition** | 4 for each target |
| **KEY digest** | sha256 |
| **CS policy** | LRU |
| **CS size** | 1000 packets |
| **CRT decisions** | * cached by edge router<br>* cached by neighbor router<br>* cached by away router |

The extracted NDN-testbed topology consists of 42 global routers (edge, neighbor, and away). To establish a VoNDN conversation, an additional 10 consumers (callee and caller) are added to each global router. Therefore, the VoNDN attack scenario is used 462 (420+42) nodes in total and 210 paired VoNDN conversations occurred.

In VoNDN, the NDN-testbed nodes are used to transmit voice/video and certificate packets to the VoNDN consumers via 42 gateway routers. In this scenario, 10 leaf nodes (callee and caller) were assigned to each edge router to represent the callee and the caller. The adversaries attack these 42 edge routers to obtain ≈60% legitimate cached certificate locations.

In the simulation, the adversaries, the callee(s), and the callers were randomly selected for each simulation run. The results were collected from in total of 20 simulation runs for each forwarding strategy (best-route and multicast) with LRU CS policy. The targeted certificates were retrieved by brute-force and each target request was repeated 4 times to increase the attack success. When the attack is finished, the adversaries compare the CRTs of certificates that were collected from the routers to decide where/when the certificates have been cached. As shown in Table 6.2, the NDN-testbed routers were classified by the following terms: *i. edge routers* represent the first-hop routers of the leaf nodes, *ii. neighbor routers* are the second hop routers of the leaf nodes, and *iii. away routers* are those located at more than two hops away from the leaf nodes. A certificate authority may be located in an away router.

**VoNDN configuration**. In the VoNDN scenario, each content (*e.g.* voice/video, media, and certificate) is signed by its callee and caller to authenticate the conversation. To verify the callee and caller public keys, the certificate authority (CA) publishes a certificate that binds the name and the public-key with a CA signature. The certificates are cached by the NDN-testbed routers to establish the call session the next time. The trusted VoNDN application scenario was implemented to analyze brute-force attack findings.

In the VoNDN application, the callee and caller are paired and send their packets at a constant rate (100 packets/s) to each other. Also, the callee and caller are configured to request a data packet, that can be presented as voice/video, media, and certificate. These can be requested by an interest packet ($\approx$50 kB). Note that, the voice conversation does not have any silence period because the callee and callers are configured with a default constant bit rate. Also, because VoNDN is implemented only for attack purposes, encoding/decoding voice and video format (*e.g.* H.264) were not implemented.

In the VoNDN scenario, each named data packets is signed by using public-key cryptography. to provide the integrity of the callee and caller. This can be trusted by a certificate authority or self-signed certificates.

On the other hand, the voice/video, media, and certificate packets can be transmitted with different routing strategies to recover the packet loss (*e.g.* due to traffic congestion). The attack scenario was implemented using two routing strategies: *i.* best route, where packets are routed through the best path between the nodes and *ii.* multicast, where packets are routed to group nodes forwarding strategies. These were implemented with least recently used (LRU), this cache replacement policy discards the least recently used certificate first from the content store.

In both applications (NDNtube and VoNDN), the CHR, CRT, and hop count metrics were collected and analyzed to evaluate the attack and the results.

## 6.2.1 Network Topology

Realistic Internet network topologies are of considerable importance to network researchers. To achieve the most realistic attack scenario, real data-set topologies were used. To do that, the NDN-testbed and AT&T network topologies were used. These topologies are extracted using the rocket-fuel data set [94]. The AT&T network topology raw data set was converted to ndnSIM format by ndnSIM rocket-fuel extension.

**Rocket-fuel mapper**. The rocket-fuel is a mapper engine that is used to convert the real-set network topology to the simulation format. Rocket-fuel extracts the backbone (weights and link latencies) using the information present in the connectivity structure of the NDN routers. Trace-route data obtained as part of topology (such as AT&T) measurement itself used as input. Rocket-fuel also can collapse the interfaces on the same router (alias resolution), leading to more accurate ISP maps. Rocket-fuel minimizes the number of paths measured between the same pair of nodes to reduce measurements required to collect an ISP map [94], [95], [96].

As a summary, the rocket-fuel can be used by *i.* To create the topology based on the real-topology. *ii.* To convert the raw topology sets to the NDN simulator format.

The following scenario topologies were simulated in this work: *i.* tree, *ii.* AT&T, and *iii.* NDN-testbed. The streaming-like (NDNtube) application was simulated on tree and AT&T topologies. The VoNDN application was only simulated on the NDN-testbed topology.

**Tree topology**. Figure 6.3 illustrated the tree topology, it is formed by sixteen consumers (called leaf), eight backbone (bb) routers, eight central gateway routers (cgw), and one producer (gw-root). Two adversary nodes (leaf-6 and leaf-13) were placed into the topology. The bandwidth of the links was 10, 100, and 1000 Mbps respectively.

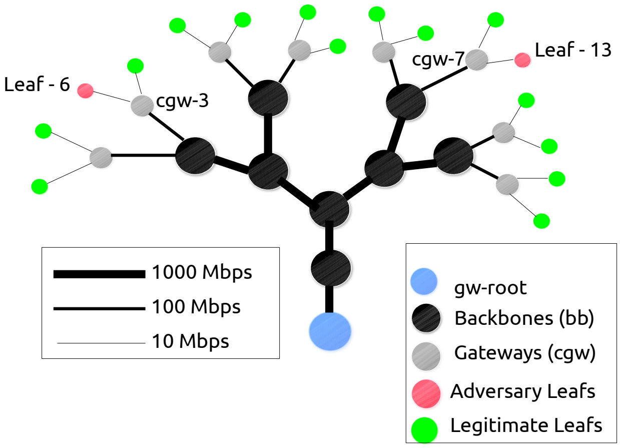


Figure 6.3: Physical tree topology of the simulation scenario.

The minimum and maximum delays of the links are presented in Table 6.3 and were obtained from the data set of a real topology. NFD was configured for best-route strategy, to have the best network paths to the consumer nodes.

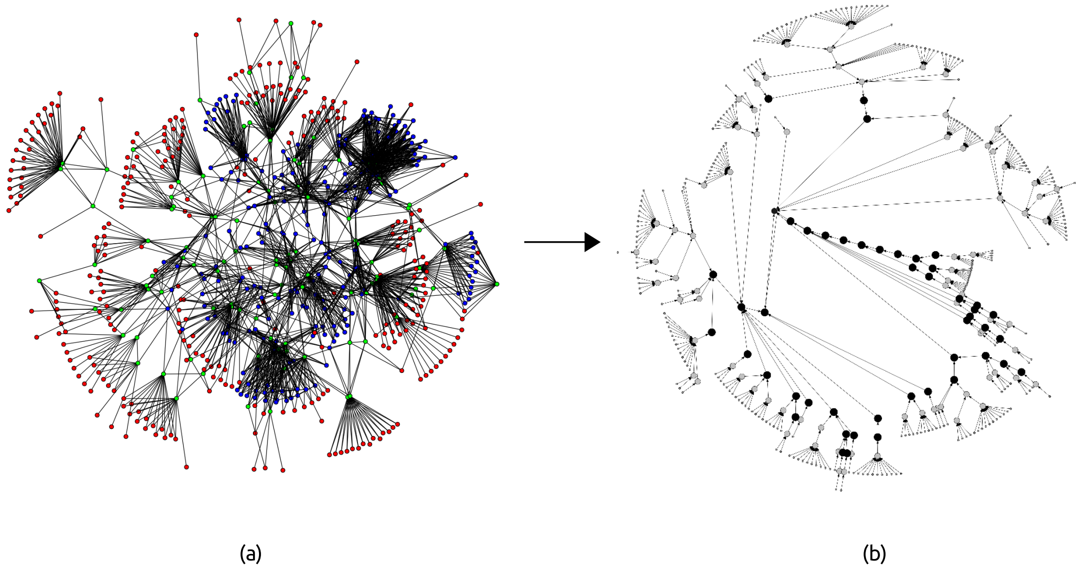
Table 6.3: Tree topology delays (min/max) between nodes linked directly.

| Delays (ms) | bb | cgw | leafs | gw-root |
|---|---|---|---|---|
| bb | 2.51 / 7.56 | 3.11/ 9.10 | - | 4.77 |
| cgw | 3.11 / 9.10 | - | 0.15 / 9.67 | - |

The adversary nodes sent interest packets at a rate of 100 packets/s with malicious interest prefixes, and the legitimate nodes at a rate of 100 packets/s sending unique (not requesting the same content name again) interest prefixes. The legitimate leaf nodes were configured to generate interest traffic with a randomized uniform (7 leaves) pattern, which is distribution in range (0, 1/Frequency) and an exponential pattern (7 leaves), which exponential distribution with mean 1/Frequency.

**ISP topology**. The map dataset of the ISP AT&T topology (Figure 6.4) was used to simulate the streaming application NDNtube. In the raw data of Figure 6.4a, the red, green, and blue nodes represent respectively the edges, gateways, and border routers. Then raw topology was converted by the rocket-fuel mapper to be readable for ndnSIM.

Figure 6.4b illustrates one of the simulations run for AT&T topology. In each simulation run, the border routers were selected randomly for the producer node, as well as the edge routers for the leaves (legitimate consumers and adversaries). In this topology, the adversary nodes attack each of the gateway router (edge) to obtain the existence of cached targets.



(a)

(b)

Figure 6.4: Real network topology conversion for NDNtube: (a) Raw AT&T network topology, (b). Rocket-fuel converted AT&T for ndnSIM.

Table 6.4 shows the bandwidth and delay values of the different types of links. In this ISP topology, the best-route forwarding strategy is selected under different CS policies (LRU, LFU, and FIFO). In the NDNtube application, all good leafs are directed to the producer, to request the data packets at a rate of 100 packets/s. The video segments are cached by gateway and backbone routers and these are presented as intermediate nodes.

Table 6.4: AT&T topology link bandwidth and delays.

| **Link Type** | **Delay** | | **Bandwidth** | |
|---|---|---|---|---|
| | min. | max. | min. | max. |
| Client-Gateway | 10 ms | 70 ms | 1 Mbps | 3 Mbps |
| Gateway-Backbone.<br>Gateway-Gateway | 5 ms | 10 ms | 10 Mbps | 20 Mbps |
| Backbone-Backbone | 5 ms | 10 ms | 40 Mbps | 100 Mbps |

**NDN-testbed topology**. The VoNDN was simulated on a real NDN-testbed topology. The testbed is consists of 42 NDN routers on the global participating institutions [14]. To understand the NDN paradigm and its instruments, the University of Minho is participating in the NDN-testbed project since 2016. Currently, the minho NDN-testbed node is operated at Computer Communication and Networks (CCN) laboratory to contribute NDN researches [97].

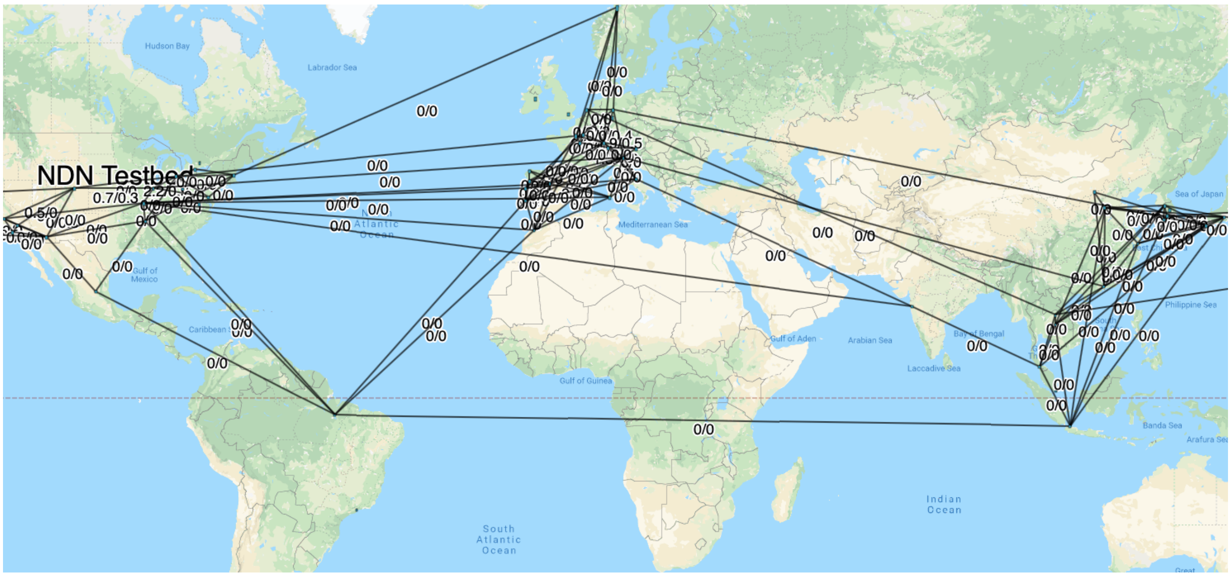


Figure 6.5: NDN testbed topology (adapted from [14]).

Figure 6.5 illustrates the current global NDN-testbed topology with its gateway routers. This topology was implemented in the NDN simulator (ndnSIM).

Table 6.5 shows the minimum and maximum delays of the links and bandwidths of the NDN-testbed. In NDN-testbed, the link delay may vary between the nodes. This is caused by a link design between nodes. For instance, minho node is only linked with basel, coruna, copelabs, urjc, and padua. Also, the callee and caller are leaf nodes of the NDN-testbed.

Table 6.5: NDN-testbed bandwidth and delays of the links.

| Testbed link type | Delay (ms) | Bandwidth (Mbps) |
|---|---|---|
| leaf-router | 1 | 1000 |
| router-router | 2-155 | 1000 |

## 6.2.2 Attack Implementation

The attack application was written in C++ according to C++11 standard and using C++11 standard libraries, and compiled with ndnSIM 2.6. In this application, the targets and attack repetitions can be formed by the adversary. Also, the application was based on a randomized function to probe the targets.

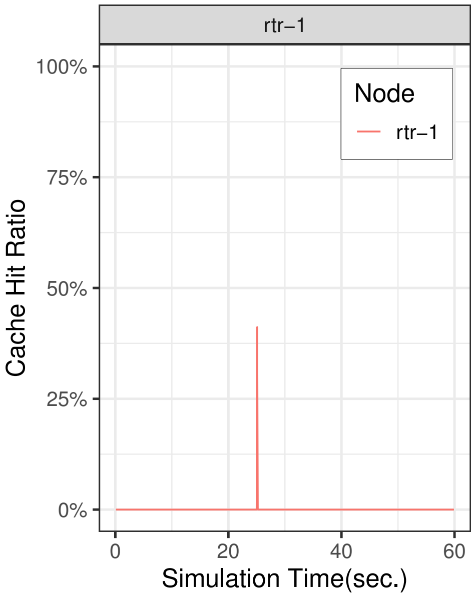


Figure 6.6: Simple attack cache hit ratio result.

Figure 6.6 illustrates the implemented simple attack scenario result for the NDNtube application. This simple scenario is consists of three nodes to illustrating the attack probability which was also presented previously (*Subsection 3.2.3*). In this scenario, the producer node issues the streamed contents for */ndntube/videos*, the consumer node (leaf-1) requests/consumes contents (*/ndntube/videos/video-1*) with consumer CBR application. Also, each of the produced contents is signed by the producer with the public key to maintaining the integrity of the content.

The produced contents (segments) are cached by an intermediate (edge) router (rtr-1). The router's CS caches the streamed contents by the size of 100 packets with the LRU cache policy (ndnSIM default). In this simple attack scenario, an adversary node was placed at side of the consumer (leaf-1) node and targeted named video segments of */ndntube/videos/video-1* and */ndntube/videos/video-2*. Note that, this scenario assumes that an adversary already had acknowledged the CRT value previously that occurs between router to the consumer.

When the attack is completed for */ndntube/videos/video-1* and */ndntube/videos/video-2*, an adversary analyzes their CRT values to conclude which one has been cached recently. Because the consumer (leaf-1) already consumed content (*/ndntube/videos/video-1*) previously, the CRT of */ndntube/videos/video-1* is slightly smaller than */ndntube/videos/video-2*. Also, an adversary retrieves one of these video content segments (*e.g. /ndntube/videos/video-1/%FE%09%C3*) which can be considered as a frame of the video.

To measure the performance of the attack, the edge router's cache hit ratio (CHR) was calculated during the attack period as previously illustrated in *Subsection 3.2.3* (Figure 3.2). In this particular attack scenario, the CHR was obtained 41% of the edge router for a single content object (Figure 6.6). In this attack design, the attack was configured by attack repetitions by four times to increase the success of the attack. But, this can be varied by the attack design.

Note that, this success is calculated by this attack design which means that the success rate can be changed by the attack design and application. For instance, a larger set CS (>100) may increase the attack success depending on the application configuration.

### 6.2.3 Attack Scope

In this work, the attack implementations are based on name privacy and certificate privacy. To illustrate that the attack scopes are defined as a content monitoring on NDNtube and a certificate-based attack on VoNDN scenario configurations.

Next, simple attack scenarios are presented to illustrate the attack findings for both applications respectively.

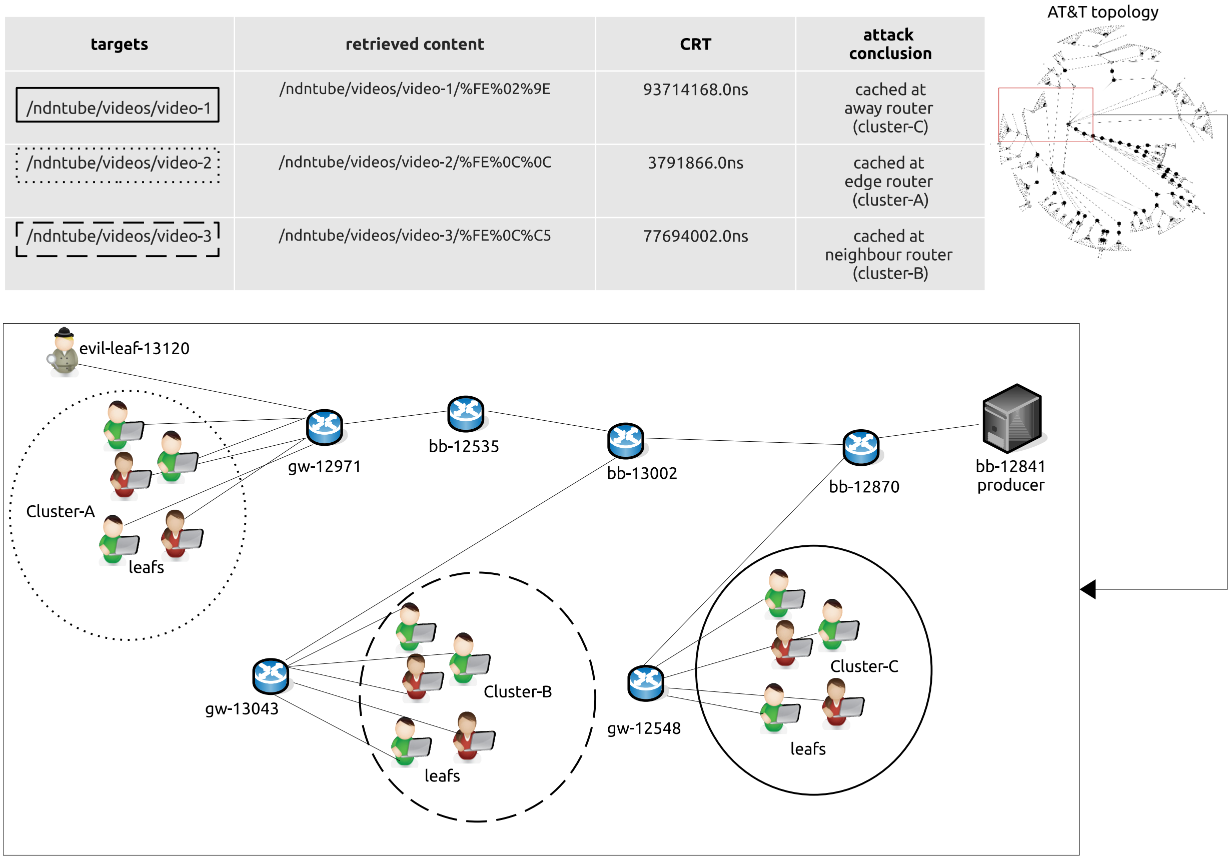
**Random node selection**. In the NDNtube-like scenario, the single producer and the backbone routers were selected from backbone (bb-) nodes of AT&T topology. In both attack applications, the all-nodes (consumers, producers, and adversaries) were selected and placed randomly in each simulation run.

In the VoNDN-like attack scenario, the consumers were selected and renamed as good-leaf, the adversary nodes were renamed and selected as evil-leaf, the gateways (edge routers) presented as gateway (gw-) nodes in NDN-testbed topology.

**NDNtube attack scenario on AT&T topology**. In NDNtube, a producer was selected from backbone (bb-) node candidates and produced */ndntube/videos* to consumers (leafs). Also, the gateway (gw-) and backbone (bb-) nodes were used to cached video segments. The adversary nodes (evil-leaf), attempt to learn video segment locations by probing the targets with brute-force and randomly. In this design, the adversaries were randomly located in the network and retrieved contents from the edge routers, which were previously requested by the leaf nodes. Also, the scenario is configured with 30 simulation runs (LRU, LFU, and FIFO). In all scenarios, the producer, leafs (consumers), and evil (adversary) nodes are randomly selected for each run.

As Figure 6.7 shows a part of AT&T topology is extracted out of 625 nodes to illustrate brute-force implementation and attack findings. In this streaming-like application, a randomly selected producer (bb-12841), consumers (leafs), an adversary (evil-leaf-13120), backbone (bb-), and gateway (gw-) were randomly selected from the number of node candidates. Also, each bb and gw is caching contents from bb-12841 with the LRU cache policy.

In this sample of attack, some targets (*/ndntube/videos/video-1*, *2*, and *3*) were defined and randomly retrieved by an evil-leaf-13120 node. For instance, if a leaf node requested content with the name (*/ndntube/videos/video-2*) then, an intermediate node or the producer replied with a video segment prefix (*/ndntube/videos/video-2/%FE%2*) to the evil-leaf-13120.



| targets | retrieved content | CRT | attack conclusion |
|---|---|---|---|
| /ndntube/videos/video-1 | /ndntube/videos/video-1/%FE%02%9E | 93714168.0ns | cached at away router (cluster-C) |
| /ndntube/videos/video-2 | /ndntube/videos/video-2/%FE%0C%0C | 3791866.0ns | cached at edge router (cluster-A) |
| /ndntube/videos/video-3 | /ndntube/videos/video-3/%FE%0C%C5 | 77694002.0ns | cached at neighbour router (cluster-B) |

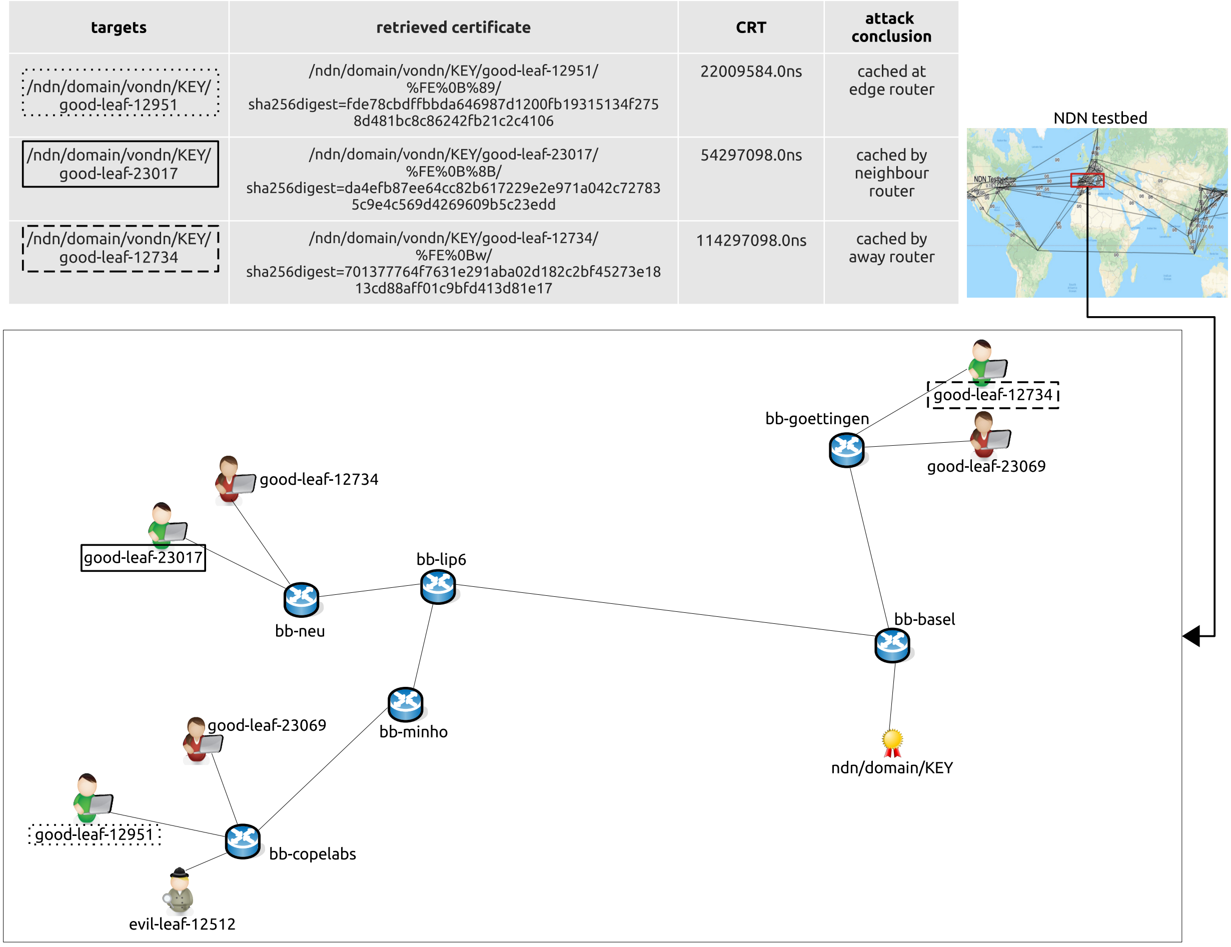
Figure 6.7: NDNtube attack scenario sample (example) on AT&T.

In traditional timing attack designs, the adversary defines the target with its prefix. Unlike other attack scenarios described in the scientific literature, our attack design uses brute force that is based on leaf names (e.g. */ndntube/videos/video-2*) and the corresponding gateway (edge router) replies with segment (/ndntube/videos/video-2/%FE%2) with lowest CRT. In this sample scenario, an adversary is also able to determine the away targets by comparing their retrieved CRTs. For instance, an adversary can assume that the target */ndntube/videos/video-3* (Cluster-B) is located close to the edge router (neighbor) and */ndntube/videos/video-1* is located to away router (Cluster-C) by analyzing their CRT differences.

In this attack scenario, the scope of the attack is considered as monitoring the popular streaming or pre-streamed content(s) by their locations. Note that, the CRT values were only used to illustrate the attack findings which can be varied by adversary location.

**VoNDN attack scenario on NDN-testbed topology**. The public key certificate is based on names, these are produced by the Certificate Authority (CA). In the VoNDN attack scenario, the trusted certificate may present the callee or the caller locations, established call time, and who established the call.

Figure 6.8 illustrates the attack demonstration which was selected from 462 nodes NDN-testbed topology. The callee and callers exchange packets these presented as certificate, voice, and SIP within call identifier. In this scenario, each of callee and caller have a unique call identifier (*e.g. /vondn/good-leaf-12625/call-*



| targets | retrieved certificate | CRT | attack conclusion |
|---|---|---|---|
| /ndn/domain/vondn/KEY/ good-leaf-12951 | /ndn/domain/vondn/KEY/good-leaf-12951/ %FE%0B%89/ sha256digest=fde78cbdffbbda646987d1200fb19315134f275 8d481bc8c86242fb21c2c4106 | 22009584.0ns | cached at edge router |
| /ndn/domain/vondn/KEY/ good-leaf-23017 | /ndn/domain/vondn/KEY/good-leaf-23017/ %FE%0B%8B/ sha256digest=da4efb87ee64cc82b617229e2e971a042c72783 5c9e4c569d4269609b5c23edd | 54297098.0ns | cached by neighbour router |
| /ndn/domain/vondn/KEY/ good-leaf-12734 | /ndn/domain/vondn/KEY/good-leaf-12734/ %FE%0Bw/ sha256digest=701377764f7631e291aba02d182c2bf45273e18 13cd88aff01c9bfd413d81e17 | 114297098.0ns | cached by away router |

Figure 6.8: VoNDN attack scenario sample (example) on NDN-testbed.

*id/2629757*). Also, callee or callee can verify each other public key by requesting a certificate that is published whether self-signed or signed by the certificate authority (ndn/domain/KEY).

In this attack, the adversary targeted three named certificates these were cached by gateway (gw-) routers. There were three VoNDN conversations between the six good-leafs and they also exchange the certificates respectively. The CRT values were compared to conclude the estimated target locations for three targets (good-leaf-12951, 23017, and 12734). In this sample, the adversary concluded that the good-lead-12951 was located the same as the adversary (edge router), the good-23017 was located at the neighbor router, and the good-leaf-12734 was located by the away router. Through this attack, the adversary can estimate, when the conversation is started and the location of the callee and caller.

## 6.3 Results

To analyze the simulated scenario results, the metrics CRT, CHR, and hop counts were analyzed. These results are collected from different CS policies (LRU, LFU, and FIFO) and forwarding strategies (best-route and multicast–under LRU). The results were analyzed based on the following: *i.* to evaluate the brute-force attack performance for multiple targets in trusted VoNDN application using the CHR results, *ii.* to analyze the attack based on the location information about the callee and the caller, *iii.* to compare DaD performance with a static countermeasure (probabilistic caching) to mitigate the brute-force timing attack, and *iv.* to analyze the performance of the content distribution between a statically configured countermeasure (probabilistically caching) and DaD by analyzing the CRT and hop count metrics.

In this work, the probabilistically caching was implemented that stands as a static router configuration to be compared with DaD. This comparison attempts show that how DaD can be an efficient approach to mitigate the attack and maintaining the legitimate requests in both NDNtube and VoNDN applications.

**Scenarios**. In both applications (NDNtube and VoNDN), the performance of the attack is measured by the obtained CHR metric on the default scenario (no-countermeasure applied). To detect an adversary's face, the detection metrics (CRT, CHR, and hop count) are presented for NDNtube and VoNDN respectively. The implemented countermeasures (`nfd:probabilistic`, `nfd:freshness` and `nfd:DaD`) results are compared/evaluated for attack mitigation and distribution efficiency.

Table 6.6: Scenario configurations.

| **scenarios** | | **VoNDN** | **NDNtube** |
|---|---|---|---|
| | | best-route | multicast |
| | | multicast | |
| default<br>(without countermeasures) | | LRU | LRU,<br>LFU, and<br>FIFO. |
| with<br>countermeasures | statically | nfd:probabilistic | nfd:probabilistic<br>nfd:freshness |
| | dynamically | nfd:dad | nfd:dad |

Table 6.6 shows the scenario settings of the results. In NDNtube, NDN CS policies (LRU, LFU, and FIFO) are applied to obtain the metrics. In VoNDN, NDN forwarding strategies (best-route and multicast) are used to obtain the metrics. The default settings presented the attack results on application scenarios without any countermeasure applied. The countermeasures are classified into two groups:

i. Static countermeasures. The static countermeasures are applied to mitigate the attack also analyze their distribution efficiency on the applications. The cache is configured by `nfd:probabilistic` within a 10% probabilistic rate to mitigate the attack on NDNtube and VoNDN. Also, another countermeasure is called `nfd:freshness` applied to NDNtube that is used to manipulate the cache responses. As presented previously (*Subsection 3.3.1*), the default freshness time is configured $1000\mathrm{ms}$ for the NDNtube application. In order to mitigate the attack, `nfd:freshness` is set by $100\mathrm{ms}$ to mitigate the attack on the default LRU scenario.

ii. Dynamic countermeasure. The `nfd:DaD` is implemented to mitigate the attack while protecting the legitimate requests (maintaining the content distribution) in NDNtube and VoNDN. In DaD, the attack checked dynamically (*e.g.* 0.5 s in NDNtube and 0.2 s in VoNDN) then multiple countermeasures are applied only to detected adversary face(s). The applied countermeasure period is defined by the characteristic of the application (*e.g.* 3 s in NDNtube and 2 s in VoNDN).

### 6.3.1 Attack Performance and Findings

In this section, the attack findings and DaD results are presented for VoNDN and NDNtube simulation scenarios respectively. The performance of the attack (attack success) was evaluated on AT&T and NDN-testbed topologies. In these experiments, the cache hit ratio (CHR) was used to measure the adversary's attack performance for the targets in NDNtube and VoNDN applications.

In both scenarios, the brute-force attack was used to obtain the cached and un-cached targets. The adversaries configured to attack the targets (100 targets/sec.) by its randomized probing function.

**Performance of attack in NDNtube**. In NDNtube, the adversaries targeted the cached video segments to monitor famous video contents. The application can be configured by different CS policies to increase content distribution. Therefore, in this experiment, the performance of the attacks was analyzed by different CS policies (LRU, LFU, and FIFO) on the gateway (edge) routers on AT&T ISP topology.

Figure 6.9 illustrates the logarithmic CHR results on different CS scenarios in NDNtube. In these scenarios, the adversaries are configured to start the attack between 20-40s. However, the attack is finished around ≈35 because adversaries finished the attack before the 40s. Also, the preparation of attack takes ≈1s, so the attacks occurred between ≈21-35s. The following average CHR values were obtained globally from all edge routers (gw-): ≈16.4% in `nfd:LRU`, ≈15.9% in `nfd:FIFO`, and ≈18.0 in `nfd:LFU` during the attack period (≈21-35s). In these scenarios, the attack results showed that an adversary can succeed more in the least frequently used policy because it keeps the famous contents in CS compared to other cache policies. Note that, the attack success can be different depending on the number of targets, quantity of adversary, topology, and CS policy.

In NDNtube, the adversaries were able to locate the targets as the following clusters: *i.* 30.4% for edge clusters, *ii.* 18.2 % for neighbor clusters, and *iii.* 51.4% for away clusters. These results were obtained by `nfd:LRU` (default) scenario without any countermeasures applied.

**Performance of attack in VoNDN**. With multicast forwarding strategy, multiple paths are followed by data packets, and contents are cached in more routers, while in best-route only the best path's routers

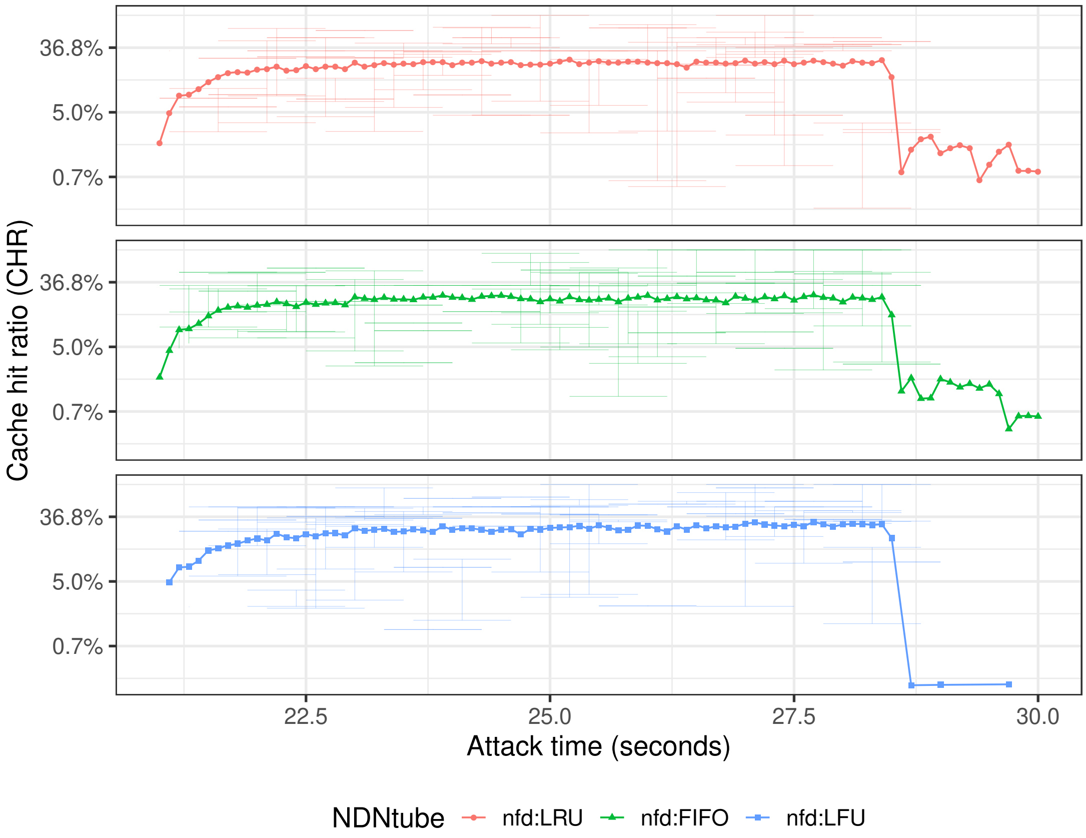


Figure 6.9: NDNtube brute-force attack performance on CS scenarios.

cache the content. In these experiments, the adversaries can distinguish between cached and un-cached targets through the retrieved CRT values and take an attack decision about the callee and caller locations.

Figure 6.10 shows the results of the CHR values to measure the performance of the attack for best-route and multicast forwarding strategies with LRU configuration VoNDN on NDN-testbed topology. In this experiment, the average of CHR was calculated globally based on all edge routers, as defined by previous *Chapter 4*, Eq. 5. The attack period was configured between 20-40s. The attack preparation takes ≈1s that is started by ≈21s. Also, the adversaries finished the attack before the 40s because the attacks were completed by ≈34s for pre-defined targets. In this scenario, the certificates were previously cached by edge routers which were used to establish a *voice/video* conversation.

The adversaries targeted the certificates to know the location of callee and caller (≈60% of legitimate nodes) by distinguishing between cached and un-cached certificates. To improve the success of the attack, the brute-force procedure can be repeated by an adversary. By accomplishing this, an adversary can distinguish between first and last repetitions. In this attack scenario, the adversary nodes retrieve the targets with four repetitions (Table 6.2). Forty percent of adversary nodes were able to target 252 certificates (60% of 420 legitimates) to identify the locations of legitimate nodes (callee and caller). The attack performance

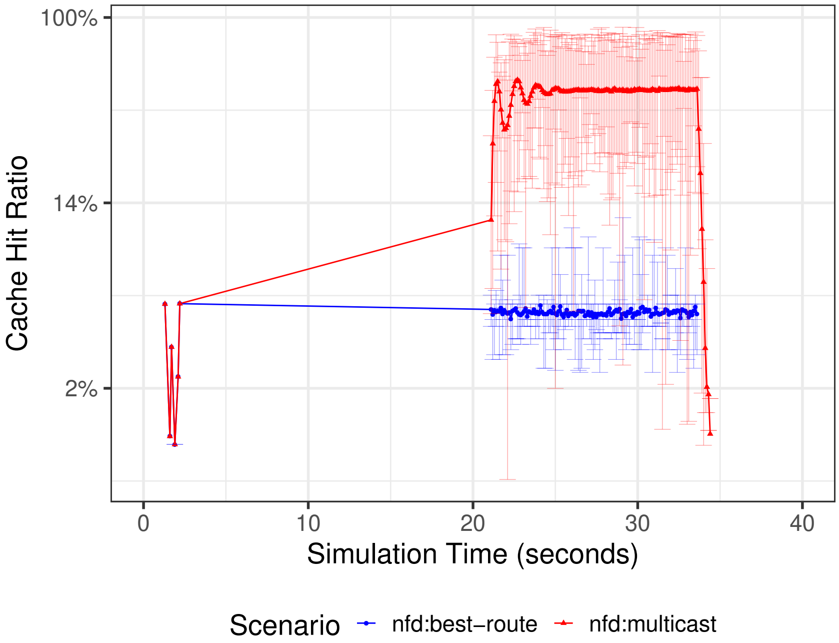


Figure 6.10: VoNDN brute-force attack performance on forwarding strategies.

measured in terms of global CHR, given by Eq. 5, presented the following values for the edge routers: ≈5% for `nfd:best-route` and ≈47.8% for `nfd:multicast`. Also, when an attack succeeds the CHR increases when not vise-versa. Thus, PID (proportional-interval-derivate) behavior occurs during the attack period.

**Certificate location determination in VoNDN**. The public key certificates are cached by the NDN-testbed routers. Since the adversary targets the consumers' certificates, these can identify the location of the consumer.

If the target (certificate) has been cached in the edge router, then the adversary hits the cache and obtains the minimum CRT. Through this attack, an adversary can identify the targets that have been cached by the edge router. Moreover, the adversary can determine the un-cached target locations by analyzing the CRT values. For instance, the maximum CRT reveals that the certificate has not been cached by any router, except by its producer (CA). If the CRT obtained is between minimum and maximum, an adversary concludes that the target has been cached by neighbor routers.

In this experiment, the adversaries were configured to distinguish the location of the cached and un-cached certificate by comparing each of the collected CRTs. The CRT values are used to classify the targets, based on three locations: *i.* cached by edge routers, *ii.* cached by neighbor routers, and *iii.* cached by away routers.

Figure 6.11 shows the results of the target locations based on the CRTs obtained in the VoNDN multicast default scenario without countermeasures. In this experiment, the adversaries were configured to target

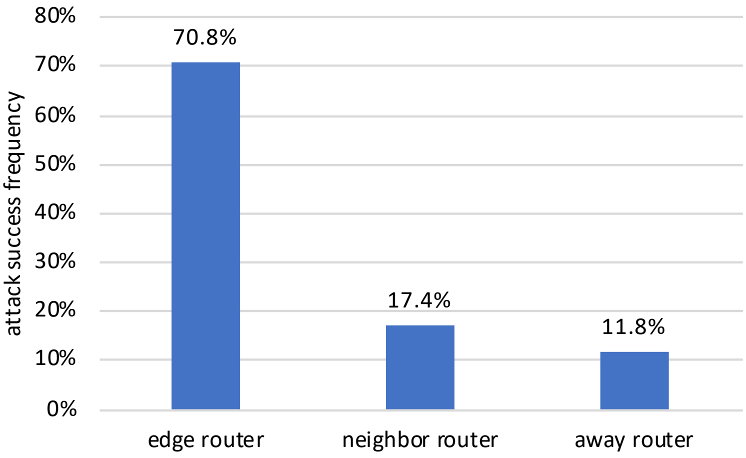


Figure 6.11: Determine certificate locations in VoNDN.

all certificates these are cached by various locations (edge, neighbor, and away). Also, the legitimate and adversary nodes are selected randomly on each attack scenario (20 simulation runs in total). The computed location findings are based on all adversary's CRT calculations. The adversaries concluded that the targets were located on the testbed routers (42 in total) as follows: *i.* 70.8% cached by the edge router (≈30 routers), *ii.* 17.4% (≈7 routers) cached by the neighbor router, and *iii.* 11.8% (≈5 routers) cached by the away routers or certificate authority in the testbed topology. In this attack scenario, the adversaries are configured to attack all possible targets at the same time. Because of this, the success of the attack is computed highest of the edge router. Also, the forwarding strategy increased the success of the attack because every node cached the certificates with multicast.

## 6.3.2 Attack Detection Results

In a side-channel timing attack, the attack purpose (scope) can be different from the attack configuration and the application. Thus, the attack detection methods and their threshold values can be different for the applications. To detect the face of the router that is attacked, the detection methods (CRT, CHR, and hop counts) were analyzed on the NDNtube and the VoNDN applications.

During the attack, each adversary repeats the targets at least four times which changes abnormally to the pattern of the metrics (CRT, CHR, and hop counts) and these variance values can be used to define a threshold for applications.

In this work implementations, the thresholds are pre-defined for NDNtube and VoNDN respectively. Also, the threshold values can be computed dynamically depending on the application and its cache policy. Note that, because of the factors (*e.g.* cache policy, attack design/scope, network topology, packet losses, etc.), the detection metric threshold values cannot be defined as a fixed value to be applicable on all NDN applications.

Next, the detection methods results are presented only to illustrate how the metrics variances changes during the attack and can be used to detect an adversary's face.

**Content Retrieval Time (CRT) detection results**. The average CRT of the adversary's face may be calculated as minimum or maximum during the attack. In this experiment, the CRT values were analyzed during the attack period on the NDNtube and VoNDN applications. These values were analyzed for legitimate and adversary nodes to understand the adversary behavior.

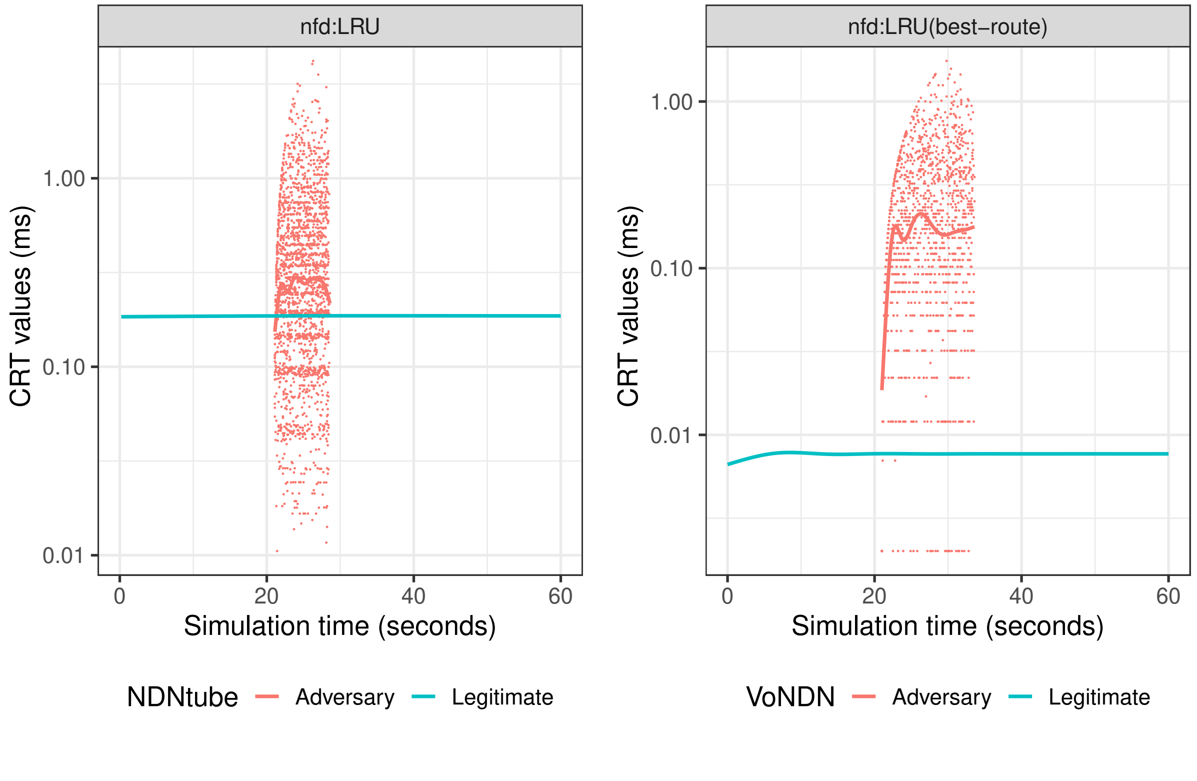
Table 6.7: Tree topology CRT Analysis.

| Leaf | Estimated CRT Threshold(s) No Attack | first CRT sample(s) | CRT Variation (%) |
|---|---|---|---|
| 1 | 0.03122045 | 0.03099957 | −0.71% |
| 2 | 0.03124 | 0.03100993 | −0.74% |
| 3 | 0.03123136 | 0.0310034 | −0.73% |
| 4 | 0.03122998 | 0.03100182 | −0.73% |
| 5 | 0.03122413 | 0.03099494 | −0.73% |
| 6 | 0.03122487 | **0.0194212** | **−37.80%** |
| 7 | 0.03122427 | 0.03099683 | −0.73% |
| 8 | 0.03122295 | 0.030993578 | −0.73% |
| 9 | 0.03123878 | 0.03101116 | −0.73% |
| 10 | 0.03123794 | 0.03100934 | −0.73% |
| 11 | 0.03123692 | 0.03100801 | −0.73% |
| 12 | 0.03123955 | 0.03101118 | −0.73% |
| 13 | 0.03123717 | **0.0144831** | **−53.64%** |
| 14 | 0.03123604 | 0.03100802 | −0.73% |
| 15 | 0.03123482 | 0.03100664 | −0.73% |
| 16 | 0.03123906 | 0.03101061 | −0.73% |

Firstly, the NDNtube was simulated on the tree topology (Fig. 6.3) to analyze the values of both the CRT samples and the CRT threshold, the scenario ran the attack and no attack periods. Table 6.7 shows the CRT threshold values were used to identify the attack of the adversarial node. The attack is detected when the CRT of the sample is below the CRT threshold. Then, we ran the scenario under attack and collected the first CRT samples. The results between the threshold and first samples are showed that CRT values may change, because of real throughput delays and congestion. However, both the expected CRT under no attack and the CRT variation (regarding the CRT threshold) reduced between ≈37% - 53% for leaf 6 and leaf 13. Therefore, these leaves are considered adversarial nodes in this particular scenario. The experimental results showed that the CRT of the adversarial leaves were shorter than the CRT of the legitimate leaves.

The second experiment was simulated using the large set topology. The NDNtube application was simulated using the AT&T (Fig. 6.4) topology and the VoNDN application was simulated using the NDN-testbed

(Fig. 6.5) topology. In these experiments, the total amount of 50% adversaries were placed on NDNtube and 40% on VoNDN.



(a)

(b)

Figure 6.12: Attack CRT detection values evaluation: (a) NDNtube global CRT values. (b) VoNDN global CRT values.

To understand the adversary behavior during the attack, the attack scenarios were configured to attack all possible targets (brute-force), these recently cached by gateway routers. Figure 6.12 illustrates the average CRT results collected from each node's faces (legitimates and adversaries). Unlike *Tree* topology, the adversarial faces CRT average was increased compared to legitimate CRT average. Because the adversary can also identify the neighbor and away routers, the maximum CRT values may present the neighbor or the away target locations.

Figure 6.12a illustrates the maximum, minimum, and average CRT values in the NDNtube-like application. The adversary concludes that the target was cached by the edge router if its CRT is less than the un-cached target(s). During the attack period, the adversary node faces CRT global average calculated 0.406 ms and the legitimate node faces CRT global average obtained 0.196 ms. The adversarial node faces CRT ≈69% increased compared to legitimate node faces during the attack. In this particular scenario, CRT values of adversaries' were obtained dramatically above from legitimate CRT. This CRT variance can be used to distinguish between legitimate and adversary nodes. For instance, the attack can be detected by pre-defined CRT threshold values in a pre-defined attack detection period.

Figure 6.12b illustrates the CRT values on each face in the VoNDN-like application on NDN-testbed. In this experiment, the CRT results were different than a streaming-like application because of the purpose of attack, cache strategy, and topology. During the attack, 0.27 ms CRT average on the adversary's faces and 0.055 ms CRT average on the legitimate node faces were measured on `nfd:LRU` scenario.

In summary, legitimate nodes CRT values computed almost stabile on NDNtube and VoNDN. During the attack, the adversary's node CRT values are computed abnormally compared to legitimate nodes. These variances are caused by the following: *i.* lowest CRT values (compared to legitimate) present targets are in cached , *ii.* maximum CRT values present the targets are cached by neighbor routers, away routers, or they are not existed (NACK packets). Thus the CRT can be used to identify the face of the node that is attacked on the edge router. Through this identification, the router can set countermeasures to adversary detected faces.

**Cache hit ratio (CHR) detection results**. During the attack, the CHR increases for the face of the adversary. This certainly occurs because of the attack repetitions for the intended target which also reveals to possible adversary face. The obtained CHR values were analyzed to illustrate the variance during the attack. In this implementation, the NDNtube was simulated on AT&T topology and the VoNDN was simulated on the NDN-testbed topology.

**NDNtube CHR results**. In NDNtube, the CHR of edge router (namely gateway) was analyzed on LRU, FIFO, and LFU cache policies. When a target was cached and retrieved four times by an adversary, the face's CHR may be increased on the edge router.

Figure 6.13 illustrated the CHR values (min., max., and average) during the simulation time (60 seconds) and the attack period is defined between $\approx$20-40 seconds. However, the attack was completed at $\approx$35 seconds because the targets were successfully retrieved before 40 seconds. In this scenario, the preparation of the attack takes $\approx$1 s therefore CHR values are analyzed after 21s. To distinguish the adversary face from the legitimate face, the CHR was analyzed during the attack period and no-attack period of the edge routers (gw-).

The multicast forwarding strategy was selected for the NDNtube with the following CS policies: `nfd:LRU`, `nfd:FIFO`, and `nfd:LFU`. The CHR values obtained by globally (from all edge routers) and following CHR average values obtained by following CS scenarios: $\approx$16.4% in `nfd:LRU`, $\approx$15.9% in `nfd:FIFO`, and $\approx$18.0% in `nfd:LFU` during attack period ($\approx$21-35 seconds). These results are showing that an adversary increases the CHR value during the attack and can be used to detect an adversary's face.

On the other hand, the CHR average values were also obtained in the no-attack period (0-21 seconds) to differentiate the adversary faces from legitimate faces. The following global CHR values were obtained by following CS scenarios: $\approx$0.03% in nfd:LRU, $\approx$0.004 in nfd:FIFO, and $\approx$0.01 in nfd:LFU policies. These values are computed lowest because legitimate nodes don't request the same content twice (expect the packet losses).

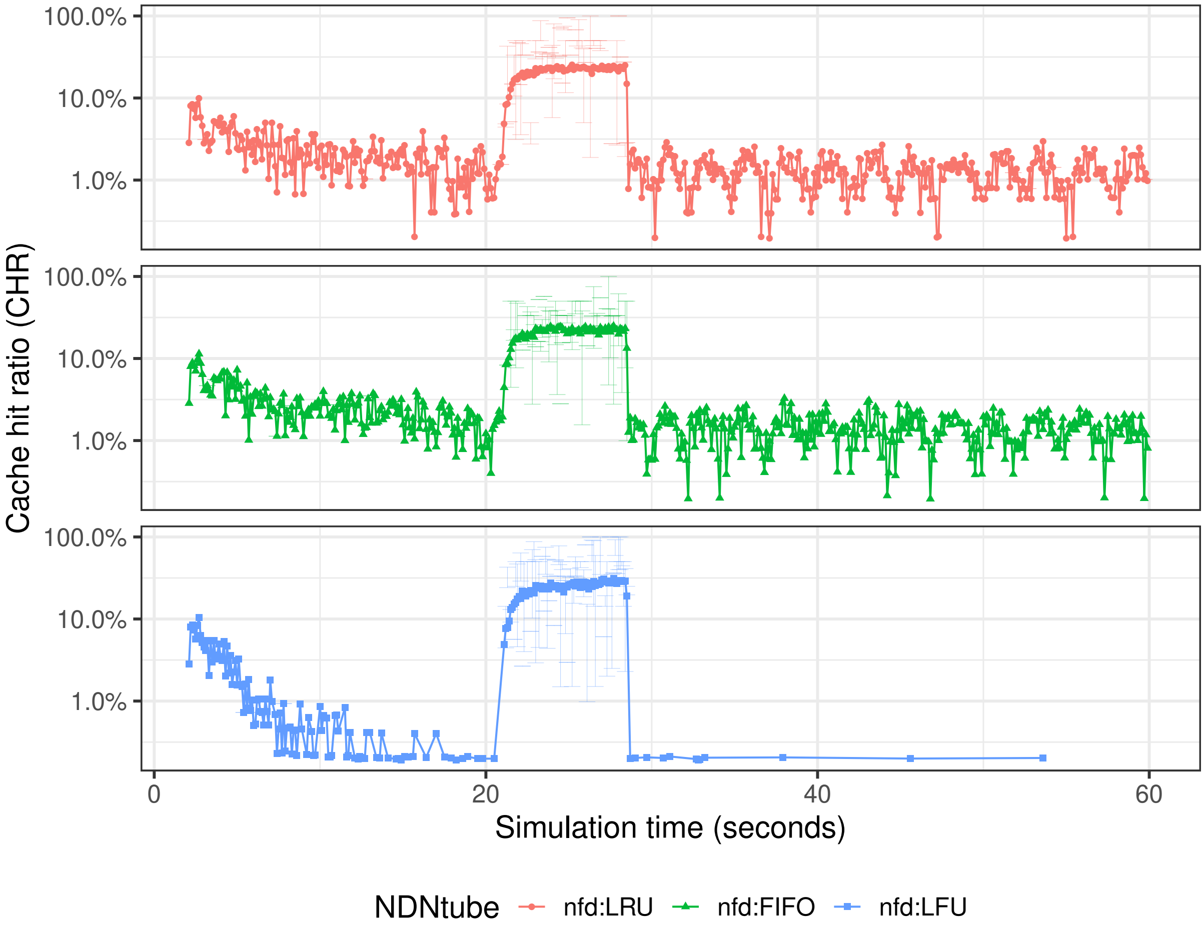


Figure 6.13: NDNtube global CHR results (edge routers).

**VoNDN CHR results.** Figure 6.14 shows the global CHR values of the edge routers on the VoNDN. The CHR values (min., max., and average–data presented in dots) were analyzed in best-route and multicast forwarding strategies in LRU. In this scenario, the callee and caller (legitimate nodes) are exchanged the certificate presented packets, at the beginning of the simulation period ≈0-5 s and the attack occurred in ≈21-35 s.

In VoNDN, the min., max., and average CHR values were obtained on `nfd:best-route` and `nfd:multicast` forwarding strategies. During the attack period, the global CHR average values were obtained on edge routers by following forwarding strategies: ≈5% in `nfd:best-route` and ≈47.8% in `nfd:multicast` during the simulation time (0-60 s).

In summary, in both NDNtube and VoNDN scenarios, the CHR noticeably increases during the attack period. Through this, a CHR threshold can be defined to distinguish the adversary faces from legitimate faces.

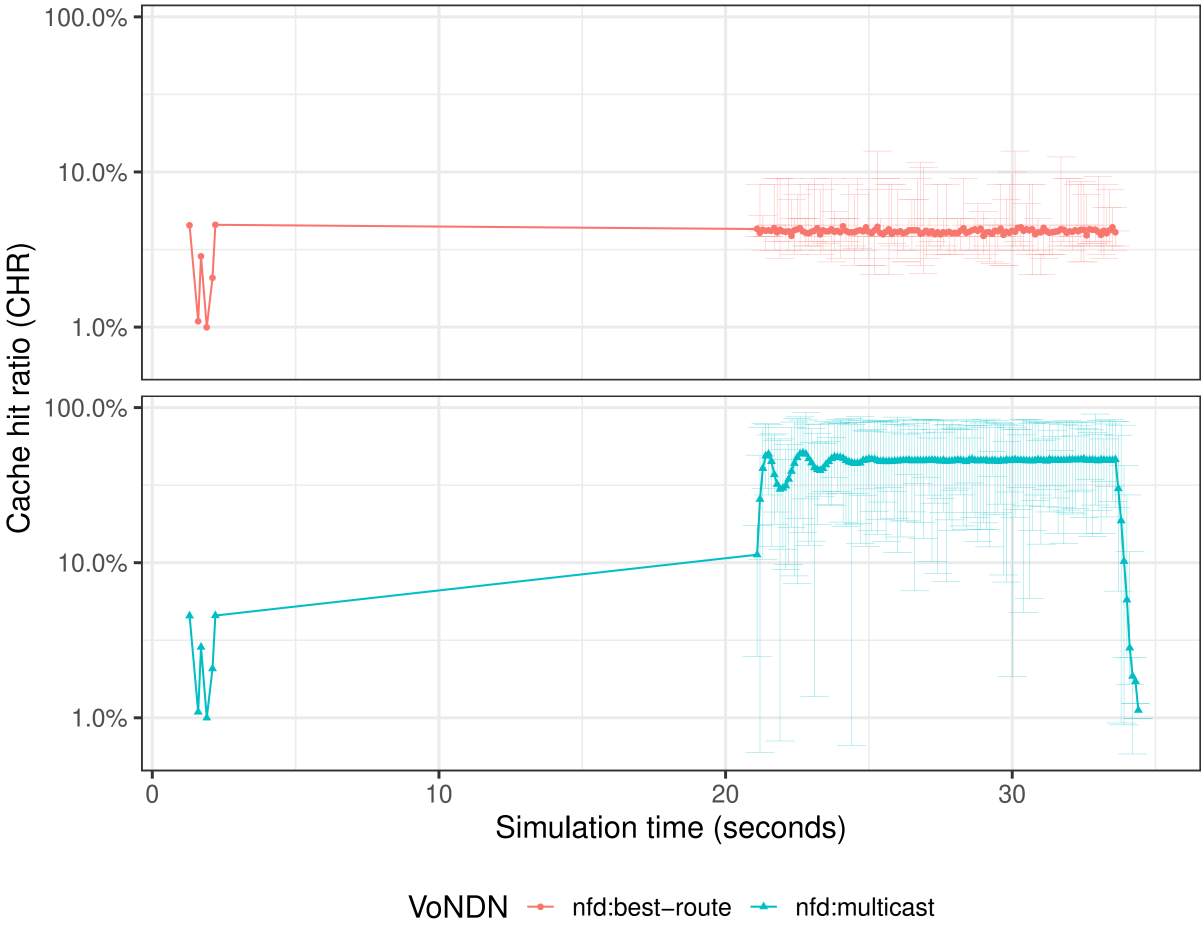


Figure 6.14: VoNDN global CHR results (edge routers).

**Hop count detection results**. Because of the attack repetitions, the hop count information is obtained lowest for the face of the router that is being attacked compared to the legitimate faces. Through this difference, a hop count threshold value can be identified to detect the adversary's face. For instance, if the target has been cached by the edge router, a hop count for adversary node obtained for "1". If not, higher hop counts occur for neighbors and away routers. In this experiment, the hop counts were analyzed on NDNtube (AT&T topology) and VoNDN (NDN-testbed topology).

**NDNtube hop count results**. In NDNtube, the adversary targets the video segments that were cached by the edge routers. Also, the adversary can obtain an approximate location of the un-cached targets comparing their CRT. When a cached video segment is cached by an edge router and targeted by an adversary repeatedly (four times), the frequency of hop count "*1*" increases for the adversary's face. Also, other hop count frequencies have occurred for the neighbor and the away routers.

Figure 6.15 illustrates the global hop counts for the adversary and legitimate node faces on NDNtube in LRU, FIFO, and LFU scenarios during the simulation time (0-60 s). In these results, the average, min., and

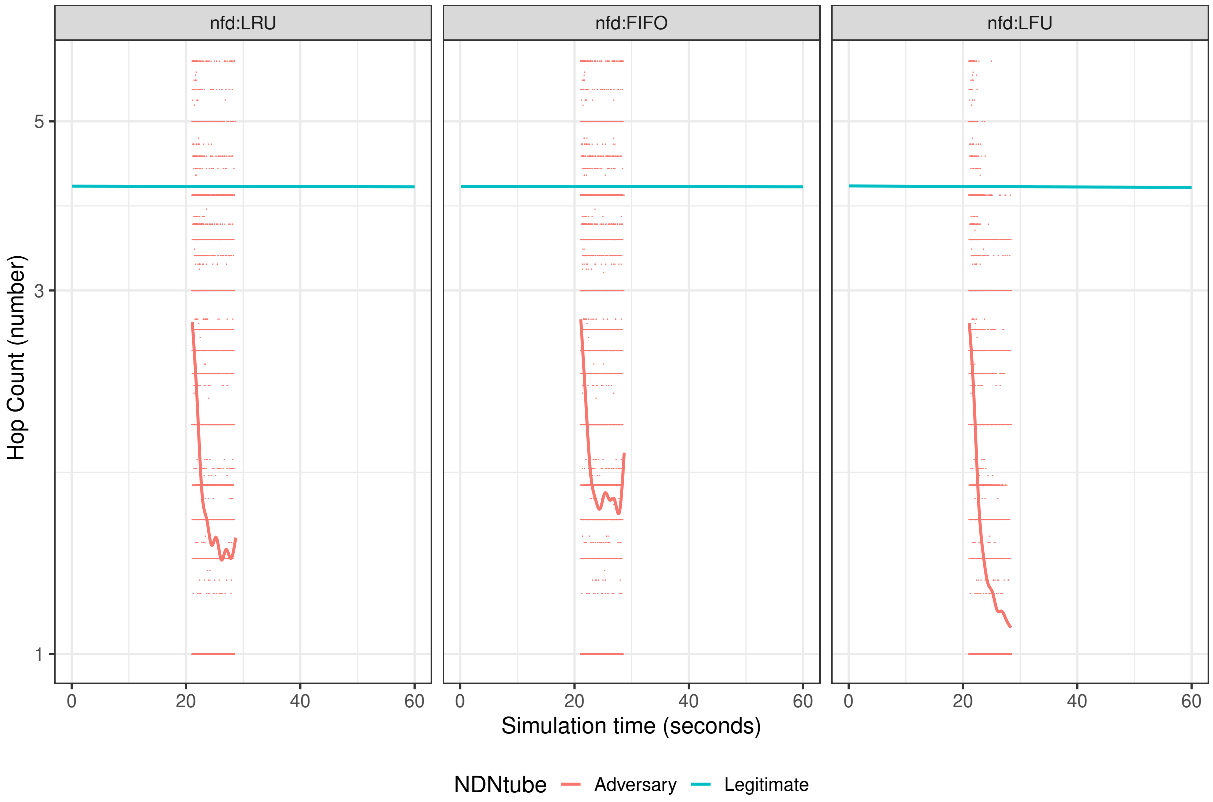


Figure 6.15: NDNtube global hop counts.

max. of hop counts were illustrated for the adversary and the legitimate faces respectively. The following average hop counts were obtained: *i.* adversary faces: 1.40 in nfd:LRU, 1.95 in nfd:FIFO, and 1.52 in nfd:LFU. *ii.* legitimate faces: 3.72 in nfd:LRU, 4.15 in nfd:FIFO, and 4.15 in nfd:LFU.

The results show that the hop count average is noticeably lower than the legitimate requests. To show this, Figure 6.16 illustrates the relative frequencies of hop counts on "*attack*" and "*no attack*" scenario simulations. In these results, the lowest hop count "*1*" presented the cached target from the first hop. For instance, with LFU, the frequency of "*1*" hop count is 20.5% under the attack period and 0.0% with no attack period.

These results show that a hop count threshold can be identified to detect the adversary face in NDNtube. As proposed in DaD, the hop counts can be useful information to detect the adversary face to apply the countermeasures.

**VoNDN hop count results**. In VoNDN, the hop counts were obtained from best-route and multicast forwarding strategies. In this experiment, the hop count metrics were analyzed globally from the NDN-testbed edge routers.

Figure 6.17 illustrates the hop count results (0-60 s) on best-route and multicast forwarding strategies in the VoNDN application. In these results, the average, min., and max. of hop counts were illustrated for the adversary and the legitimate faces respectively. In the attack period, the adversary nodes targeted all

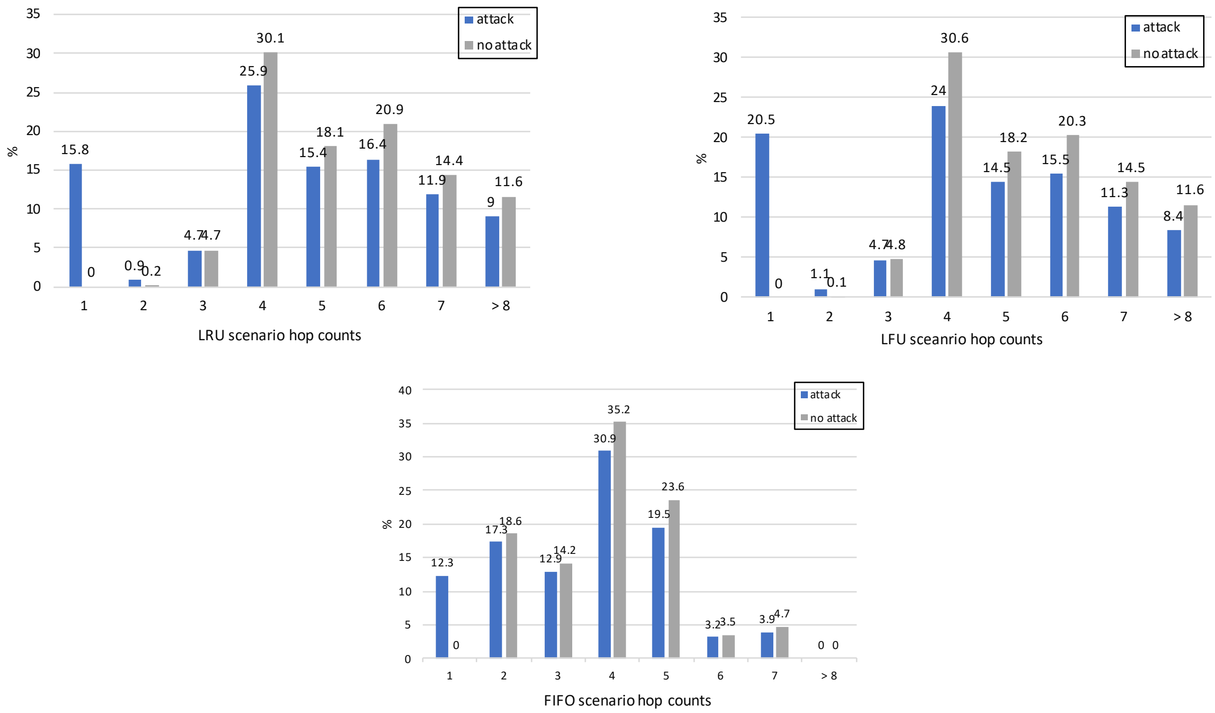


Figure 6.16: Relative hop count frequencies on NDNtube.

possible targets to retrieve information about where, when, and who requested them. Because adversaries have targeted all certificates, the best-route hop count result is calculated 1 for the adversary nodes. However, the adversary nodes have more information about other cached targets in a multicast scenario which can be used to identify neighbor and away targets.

The average hop count results (0-60 seconds) were obtained by the following: *i.* adversary faces: $\approx$1 in nfd:LRU(best-route) and $\approx$1.04 in nfd:LRU(multicast). *ii.* legitimate faces: 2.43 in nfd:LRU(best-route) and nfd:LRU(multicast) scenarios. These results showed that the average of hop counts was obtained lowest for the adversary's faces compared to the legitimate faces. Thus, a hop count threshold can be defined to detect the adversary's faces in VoNDN.

**Discussion**. To show that the adversary nodes can have abnormal behavior from legitimate nodes, the detection methods were analyzed during the attack and no-attack. The CRT, CHR, and hop count experiments were evaluated. These experimental results showed that the adversary certainly reveals itself during the attack because attack repetitions and detection methods can be used to detect the adversary's face.

To detect an adversary's face a single detection method can be used. However, each detection threshold value must be computed to be properly tunned for a certain NDN application. On the other hand, the adversary can learn information in trusted applications (*e.g.* where, when, and who), when the adversary targets the certificate privacy. Therefore, more privacy-worried trusted applications (*e.g.* certificate privacy),

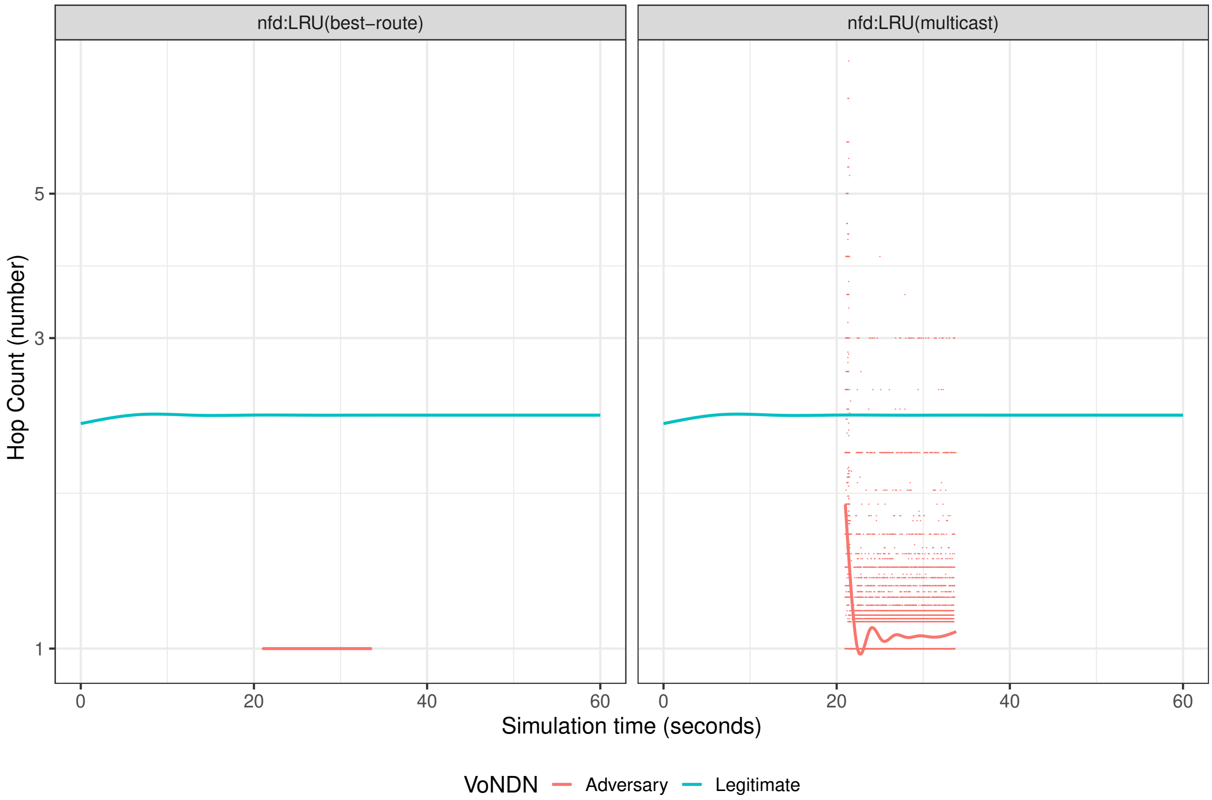


Figure 6.17: VoNDN global hop count results.

a combination of all the three (CRT, CHR, and hop count) could provide a better decision to detect an adversary's face. The purpose of the detection phase is to have a clear a precise decision on a simple matter "*face is under attack*" or face is "*not under attack*".

### 6.3.3 Countermeasures

To mitigate the attack, the countermeasures based on static probabilistic and DaD were configured with the NDN forwarding daemon (NFD), which was used as a network forwarder.

As previously introduced (*Chapter 4*), the DaD algorithm can be based on various attack detection metrics such as CRT, CHR, hop count, and name prefix. In this implementation, the DaD only configured by CHR detection with its pre-defined moving average threshold to detect an adversary on NDNtube and VoNDN.

The following CHR threshold values were identified by the following applications: *i.* In NDNtube, the CHR threshold is identified as 5% CHR to detect the adversary for the LRU scenario. *ii.* In VoNDN, CHR threshold values are used as 1% for best-route and 5% for multicast forwarding strategies. The threshold values are calculated based on legitimate requests without attack for both applications. For instance, the NDNtube consumers using the cache more than the VoNDN consumers because of the NDNtube streamed content requests.

Next, collected countermeasure results based on static probabilistic caching and DaD are presented and compared to mitigate the attack on NDNtube and VoNDN application respectively.

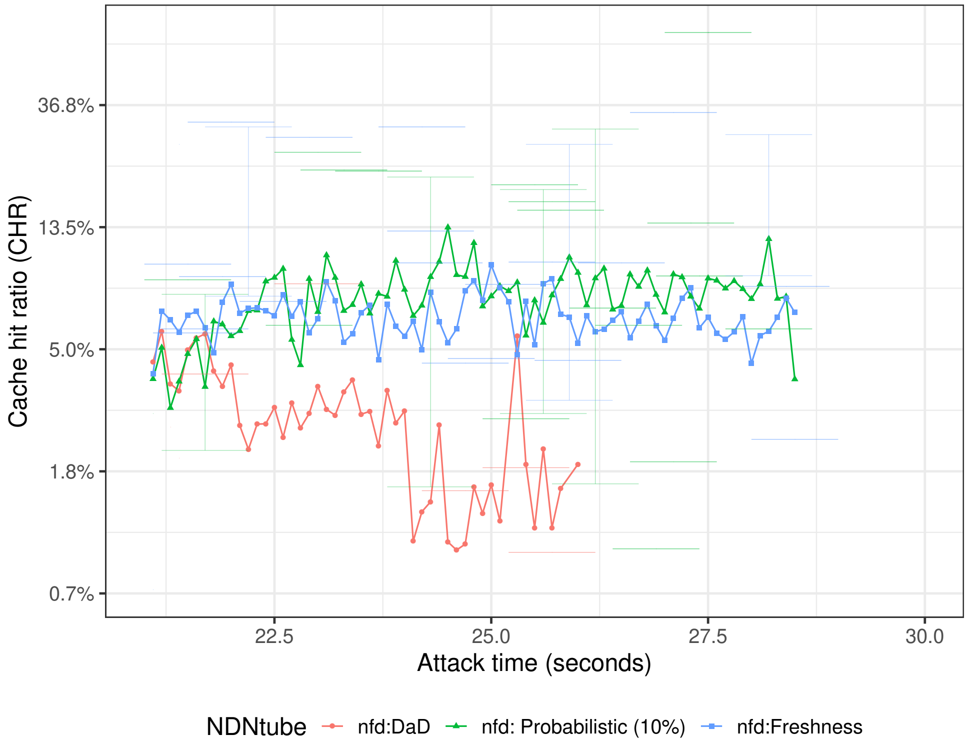


Figure 6.18: NDNtube attack mitigation results in LRU cache policy.

**Applied countermeasures on NDNtube**. In NDNtube the video segments can be also cached by the freshness period of the segment. Additionally to probabilistic caching, the segments were configured by the freshness period (≈100ms) to mitigate the attack. In NDNtube, the DaD threshold (CHR) configured as 5% to detect the adversary and the detection period configured as 0.5 seconds and applies each attack phases (minor, moderate, and severe) for 3 seconds (*Subsection 4.3.2*).

The DaD dynamically detected the attack and took countermeasure actions instead of statically configured routers to mitigate the attack. Figure 6.18 illustrates that the DaD was dynamically mitigated the attack in NDNtube and CHR was obtained 0.7% in attack period with its three phases (no-cache included). This result also illustrated that all DaD countermeasures were applied to adversarial faces because the attack was considered as severe. Also, the statically applied (for all faces) countermeasures CHR results were obtained by the following: *i.* ≈4.1% in `nfd:probabilistic` and *ii.* ≈3.7% in `nfd:freshness` scenarios.

In no-countermeasure applied nfd:LRU policy, the average CHR is obtained ≈16.4% (*Subsection 6.3.1*). This CHR decreases under applied countermeasure configurations (DaD, probabilistic, and freshness). The countermeasure results are shown that the average CHR obtained minimum (≈0.7%) in the DaD cache configuration. This may also show the legitimate requests are protected. On the other hand, the statically configured countermeasures (probabilistic and freshness) are decreased the CHR compared to no-countermeasure applied CHR. However, these countermeasures (probabilistic and freshness) were not performed effectively (≈4.1% and ≈3.7%) compared to DaD (≈0.7%) to mitigate the attacks.

**Applied countermeasures on the VoNDN**. The countermeasures were implemented in VoNDN with the best-route and the multicast forwarding strategies on the default LRU scenario. The following configurations were used to mitigate the brute-force attack:

1. The edge routers were statically configured with a probabilistic caching of 10% of content cache acceptance by randomly chosen of the data packet that can be cached. The global CHR results (adversary's faces) of these edge routers were analyzed and compared with the LRU, which is used replacement policy in ndnSIM.

2. The edge routers were configured within a DaD algorithm, which identifies the face of the router that is being attacked by checking the CHR threshold every 0.2 seconds and applying each countermeasures phase during 2 seconds (*Subsection 4.3.2*). When an attacked face is detected, the DaD applies countermeasure strategies, depending on the severity of the attack. To detect the face that is being attacked and apply countermeasures, the CHR threshold values were used. In this work, the threshold values were identified only for this particular attack scenario which may be different on other NDN applications. Through the VoNDN simulation experiences, a predefined CHR threshold was identified by moving average values as 1% CHR for best-route and 5% CHR for multicast forwarding strategies. If the attack was withdrawn by an adversary or does not exist, the DaD applies the default (LRU) phase.

In VoNDN, the implemented DaD checks the existence of an attack on the faces every 0.2 seconds and the countermeasure phases (each for 2 sec.) is only applied to the attacked edge routers, to protect the legitimate certificate requests from the edge router(s). The DaD detects the attack by checking the CHR (Eq.7 with a $\alpha$=0) threshold on every face.

Figure 6.19 illustrates the VoNDN CHR results obtained with a brute-force attack, considering the use of the probabilistic caching (10%) and the DaD in the edge routers. In both cases, the best-route forwarding strategy was used. An average CHR of 0.69% was obtained in the attack period with the probabilistic caching, which mitigated the attack ≈30.3% when compared with the results of the default LRU scenario (Figure 6.10). On the other hand, the DaD detects the attacked router first then applies different countermeasures phases while the attack persists with a 1% CHR threshold. If no attack is detected, then DaD sets the router face to the default phase. The average CHR obtained was 0.42%, which represents attack mitigation of ≈57.5%, when compared to the default LRU best-route scenario. The DaD also performed 39.1% improved the attack mitigation compared to the probabilistic caching configuration.

Figure 6.20 shows the CHR results obtained in a multicast forwarding strategy in VoNDN. The discontinuities seen at 21s, 24.5 s, and 27.5 s in the DaD graphics of Figure 6.20 (and Figure 6.19) are due to the

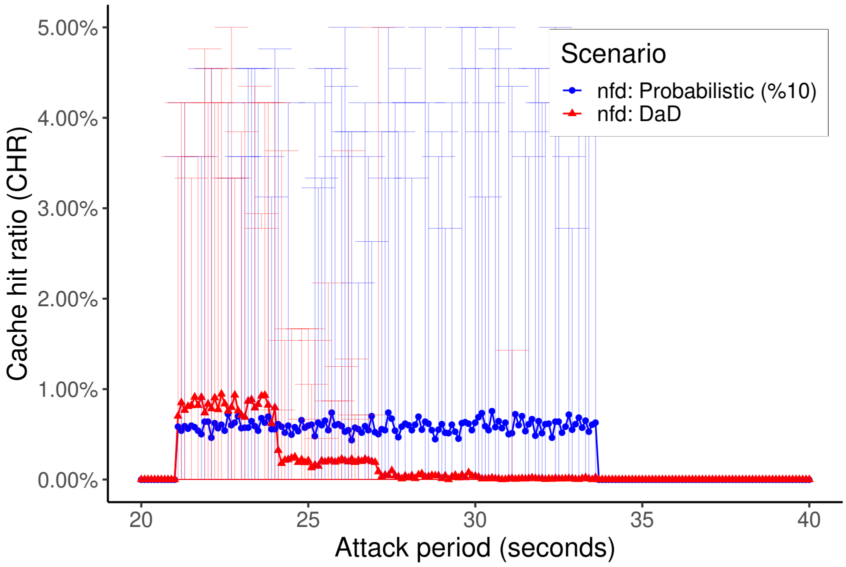


Figure 6.19: Comparisons of applied countermeasures in VoNDN best-route forwarding strategy.

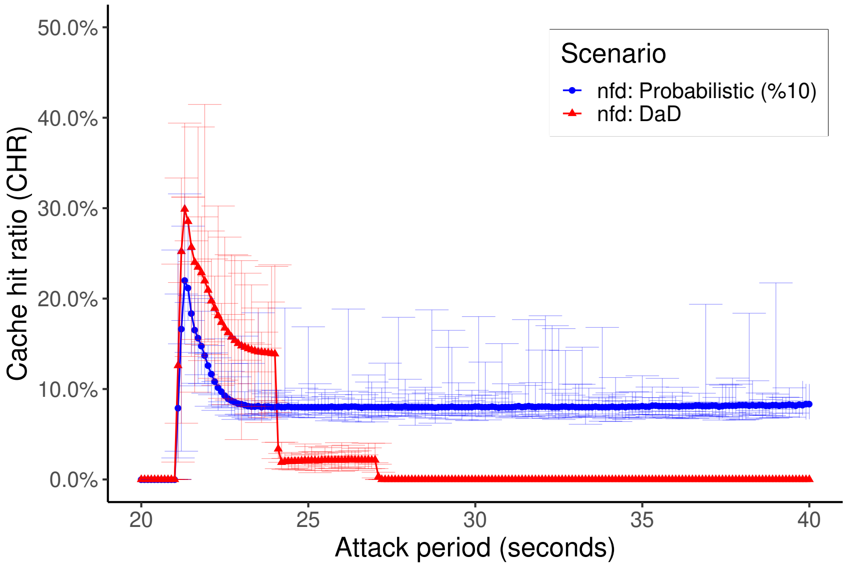


Figure 6.20: Comparisons of applied countermeasures in VoNDN multicast forwarding strategy.

application of the countermeasures. Using the probabilistic caching (10%) for all faces, an average CHR of 8.12% was obtained during the attack period. This configuration mitigated the attack by about 83.9% when compared with the default LRU scenario in multicast (Figure 6.10). The CHR threshold in DaD was configured to 5% for a multicast attack scenario. In this case, an average of CHR 5.15% was obtained, which mitigated the attack 89.8%, when compared to the default LRU multicast scenario (Figure 6.10), and mitigated the attack 36.6%, when compared with the probabilistic caching.

**Countermeasures distribution efficiency evaluation**. The DaD only applies the countermeasures to the attack detected faces instead of setting countermeasures to all faces. Thus, legitimate requests and privacy can be preserved by the DaD. To show this, the CRT (best-route) and hop counts (multicast) were analyzed on default (LRU) and countermeasures (Probabilistic and DaD) VoNDN scenario.

**Countermeasure distribution efficiency CRT results**. Figure 6.21 illustrates the global CRT results (best-route) for both adversary and legitimate nodes considering on default LRU (best-route), probabilistic caching, and DaD scenarios during the attack time (≈21-35 s). If the target is cached by the edge router, the minimum CRT values are obtained for adversary nodes otherwise it obtains increased CRT values for neighbor and away targets. To show the countermeasures (probabilistic and DaD) results to mitigate the attack on the default (LRU) scenario, the CRT values were analyzed. When the probabilistic and the DaD were applied, the adversary node's CRT value increases for the targets which are illustrating the attack mitigation rate. In this case, the adversary may not able to identify the location of the cached target because of unsteady collected CRTs.

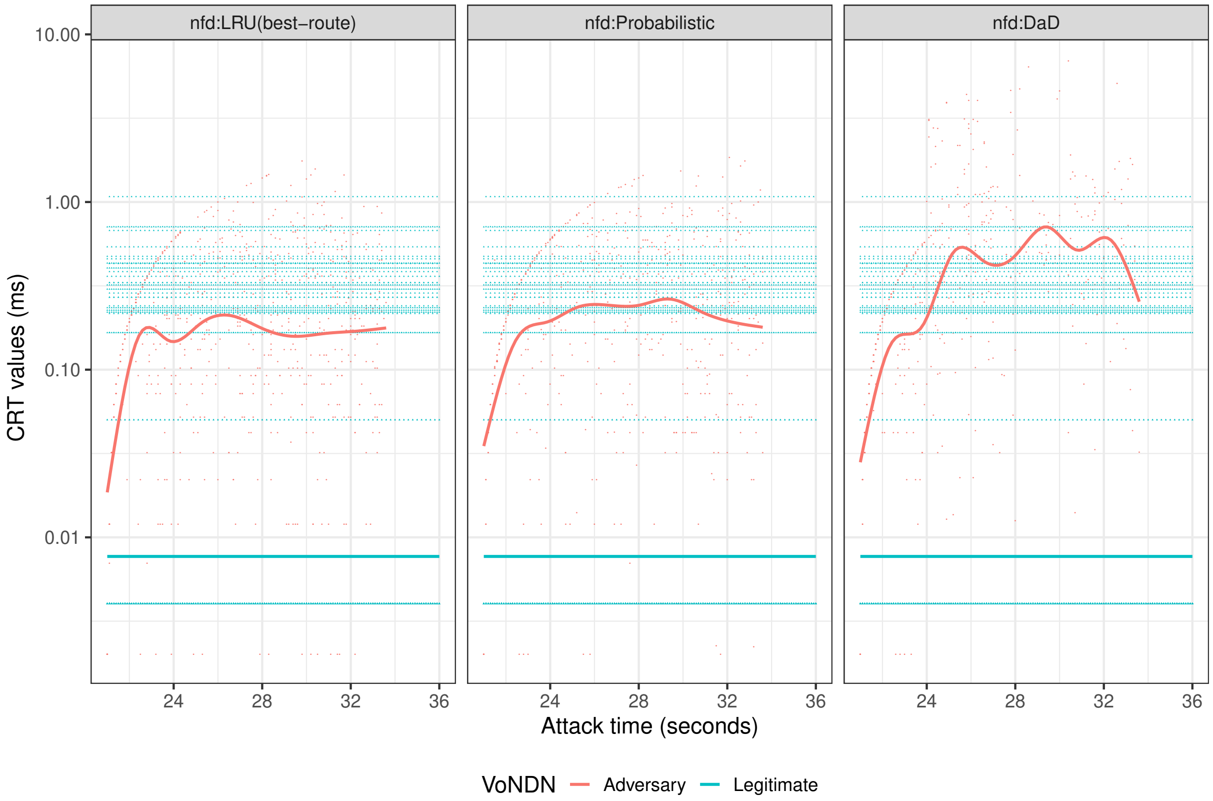


Figure 6.21: VoNDN Global CRT results for the adversary and legitimate nodes.

The countermeasures (probabilistic and DaD) can be used to mitigate the attack. However, the CRT results showing that the static probabilistic caching also increases the legitimate node CRTs, which reduces the content distribution performance for the legitimate nodes. To preserve the legitimate nodes' requests, the DaD applies the countermeasures only to the face that being attacked. Thus, an average of CRT is calculated as the same (0.056 ms) for `nfd:DaD` and `nfd:LRU (best-route)`.

Table 6.8: VoNDN CRT average values for legitimate and adversary nodes.

| scenarios | CRT average (ms) | |
|---|---|---|
| | legitimate | adversary |
| LRU (best-route) | 0.056 | 0.270 |
| probabilistic | 0.093 | 0.328 |
| DaD | 0.056 | 0.419 |

Table 6.8 shows, the CRT metrics these illustrated in Figure 6.21 in VoNDN. During the attack period, the legitimate CRT metrics were preserved by the DaD, compared to the probabilistic caching. Also, the CRT values of the adversaries are higher than those of the legitimate nodes because the adversaries were also targeted to un-cached certificates to obtain neighbor and away routers. Because DaD only applies the countermeasures to the adversary's faces, the average CRT of DaD calculated 0.419 ms which is higher than the CRT of probabilistic (0.328 ms). Thus, DaD mitigated more attack than static probabilistic countermeasure while protecting legitimate requests.

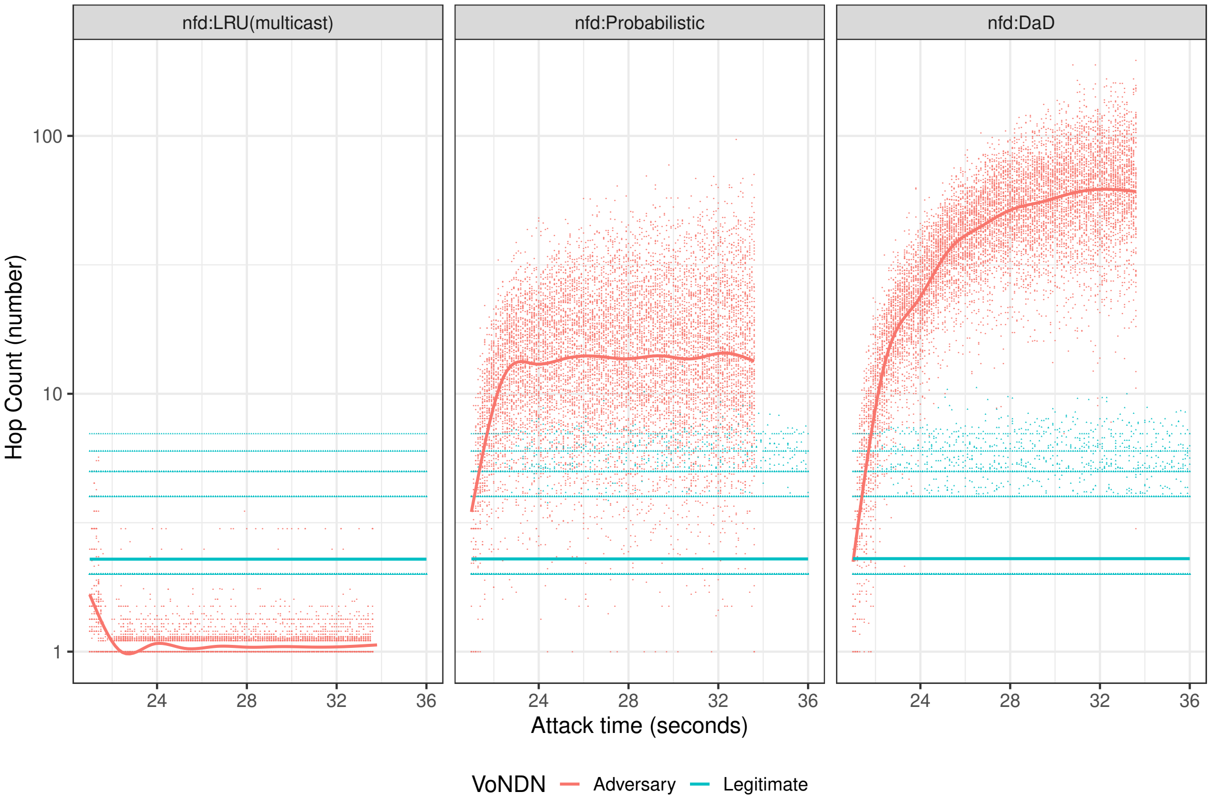


Figure 6.22: VoNDN adversary and legitimate nodes hop count metrics.

**Countermeasure distribution efficiency hop count results**. Figure 6.22 shows the global hop-count results on default LRU (multicast), probabilistic, and DaD scenarios during the attack time (≈21-35 s). If the attack is successful, the minimum hop count metric can be obtained because of adversary hits the edge routers. If an attack is not successful, the maximum hop count metrics obtained. Because of no-countermeasure applied in LRU (multicast) scenario, the adversary's average of hop-count metrics is obtained minimum (≈1.04) than the average of legitimate node's hop counts (≈2.38) during the attack.

To mitigate the attack on the default (LRU) scenario, the countermeasures (probabilistic and DaD) were set. When countermeasures are applied, the hop count of adversaries increases. However, the hop counts of legitimate nodes also increase in probabilistic caching configuration, because of its set to all faces. On the other hand, the DaD only set the countermeasures to attack the detected face, which preserved the hop counts of legitimate requests.

Table 6.9: VoNDN hop count countermeasure results.

| **scenarios** | **hop count (average)** | |
|---|---|---|
| | legitimate | adversary |
| LRU (multicast) | 2.38 | 1.04 |
| probabilistic | 2.43 | 1.84 |
| DaD | 2.38 | 35.9 |

Table 6.9 shows the average hop count metrics for the applied countermeasures (probabilistic and DaD) to mitigate the attacks on the default multicast (LRU) VoNDN scenario. During the attack period (21-40 s), the default scenario presented an average hop count of 2.38 for legitimate nodes and 1.04 for the adversary nodes.

The probabilistic and DaD scenarios are applied to mitigate the attack on the default scenario. In the probabilistic scenario, the average of adversaries hop count increased to 1.84 from 1.04 (default). This reveals the attack mitigation of about 55% on the default scenario. However, the probabilistic caching also increased the average hop counts of the legitimate certificate requests from 2.38 (default) to 2.43. Therefore, 2% of the VoNDN conversation traffic between the callee and the caller may be considered as affected or delayed because of the probabilistic scenario.

In DaD, the legitimate requests preserved and the average hop count was equal to the value obtained in the LRU (default) scenario (2.38). These values suggest that DaD may have applied the countermeasures only to the attack detected faces. In DaD, the average hop count was increased to 35.9 from 1.04 for the adversary nodes. These results are showing that the performance of attack is significantly decreased under DaD's multiple countermeasure configuration.

## 6.4 Source Code

All scenarios were scripted by the C++11 library in ndnSIM 2.6. The scenario implementations and required tools can be publicly accessible at the author GitHub account— https://git.io/fjZjZ.

## 6.5 Discussion

To understand the side-channel timing attack findings and detection the NDNtube-like and the VoNDN-like applications are developed respectively. These applications are simulated on topology sets (AT&T and NDN-testbed) to obtain realistic scenario findings.

The brute-force attack performance was evaluated for NDNtube streamed segments on the tree and AT&T topologies. The CHR was used to evaluate the performance of the attack on NDNtube CS policies (LRU, LFU, and FIFO). In NDNtube, the adversaries succeeded to obtain the popularity of targets by following clusters: edge, neighbor, and away clusters under by `nfd::LRU` (default) scenario. These results showed that the adversary may reveal information from cached CRT value, especially where the streamed content is cached to identify the popularity of contents in NDNtube. Identifying the popularity can be critical for network privacy and security. For instance, an adversary can target the popular contents to make them unavailable by requesting unpopular contents to the cache (similar sense of cache poisoning).

VoNDN attack scenario was evaluated on the NDN-testbed topology. In this scenario, the adversaries targeted 252 certificates to know the callee or caller locations. The adversaries were able to know the locations (edge, neighbor, and away) of certificates based on CRT analysis. The cached certificate may reveal such information about the conversation (*e.g.* who, when, and where) in the trusted-VoNDN application. In the VoNDN scenario, the attack results were analyzed on NDN forwarding strategies (best-route and multicast) respectively. Because the certificates are available for each callee and caller, the attack results showed that the callee and caller locations can be identified by the side-channel timing responses of the cache.

To detect an adversary's face, the detection methods were analyzed on NDNtube and VoNDN applications respectively. The metrics of CRT, CHR, and hop count are analyzed of the under attack and no-attack periods. These results showed that the metrics (CRT, CHR, and hop count) of the adversary's face changed abnormally compared with the legitimate faces. More specifically, because of the attack repetitions, the CHR increases for the adversary's faces and this can be certainly used to detect an attack. However, the other metric results are also presented and these can be used to detect an adversary's faces on the applications.

To mitigate the attack the countermeasures were developed on NDNtube and VoNDN. A static countermeasure probabilistic (`nfd::probabilistic`) cache evaluated for NDNtube and VoNDN. Additionally, another static countermeasure (`nfd:freshness`) only applied to the NDNtube. The results showed that a static method may not be the most effective to mitigate the attack. Also, the results showed that a statically configurable approach affects legitimate requests and content distribution efficiency. Then this work's main approach (`nfd::DaD`) is tested for both mitigation and distribution efficiency on NDNtube and VoNDN. The mitigation results showed that the mitigation is improved by DaD because of its multiple countermeasure configuration. Also, it is shown that the multiple countermeasures only applied to adversary detected face because of its detection configuration.

Note that, the scope of attack and detection thresholds were identified only for NDNtube and VoNDN applications. In this work, different cache policies and forwarding strategy configurations are used to show differences in the detection threshold and method. Thus, the detection methods and their threshold value can be re-adapted depending on the NDN application and cache policy.

# 7

# CONCLUSIONS

In this chapter, the work conclusions are presented. The work's main objectives and experimental objectives are also reviewed. Also, possible future works and considerations are presented.

## 7.1 Summary of Thesis

The growth of network devices and ubiquitous interconnectivity are forcing the Internet to be overwhelmingly used for content distribution. NDN paradigm attempts to recover this issue to maximize the content distribution with the in-network caching feature to answer today's application needs. The NDN features on packet types, application layers, and protocols literature were surveyed in *Chapter 2*.

Besides the benefits of caching, the previously cached contents may be faced with an attack called side-channel timing differences. The possible scope of the attack on the VoNDN and NDNtube scenarios and possible countermeasure methods were surveyed in *Chapter 3*. However, the countermeasure methods presented by other works may be considered a trade-off between privacy and the efficiency of content distribution. Therefore, this issue had motivated this work, to identify the main objective as *to propose an efficient approach to mitigate the side-channel timing attacks and maintain the content distribution in NDN*.

The main objective of the work led to focus the adversarial face detection methods. Through dynamic adversary detection, the countermeasure methods can be only applied to adversary's faces to maintain legitimate requests and content distribution. To achieve this goal, multiple countermeasures based on detection methods (mainly: cache hit ratio, hop count) were proposed by a privacy model called DaD, were it presented in *Chapter 4*.

On the other hand, an attack model called brute-force was developed to meet with today's attack trends. Through, brute-force, the adversary may identify multiple targets to increase the success of the attack.

To differentiate the attack behavior on the applications, two NDN simulation scenarios (NDNtube and VoNDN) prototypes were implemented. Also, taking into consideration the different scope of an attack, DaD, and attack findings results was presented. The results showed that the DaD can be used to maintain the legitimate requests and content distribution of both NDNtube and VoNDN simulation scenarios.

## 7.2 Reviewing Objectives

The work and experiment objectives were identified in *Chapter 1*. This section reviewed the achieved work and experimental objectives also answering the research question of this work. The objectives were reviewed by the following:

**i. Survey the NDN architecture and cache privacy-related issues.**

This objective was carried out in *Chapter 2* with a bibliographic review about NDN. Also, the content-centric network designs and features are introduced such as security, routing-forwarding, and transport functions. The in-network caching design also revised to improve the scalability, cost reduction, network performance while increasing the privacy of content. In this chapter, the NDN privacy threats were pointed out about the cache, content, name, and signature (certificate).

**ii. Survey the side-channel timing attack and its countermeasure methods to preserve content privacy.**

After identified the NDN cache privacy threats, an attack-type called side-channel timing and its countermeasures were surveyed in *Chapter 3*. It was shown that the attack scope can be used to identify the previously cached contents in NDN, these names are used to obtain the producer and consumer locations. Also, usable countermeasure statically configured methods were discussed which were presented by other researchers. To illustrate the countermeasures efficiency on content distribution, countermeasures are classified mainly based on cache available and unavailable methods. It is also discussed why the statically configured countermeasures can be inefficient when considering the distribution of content in NDN.

**iii. Understand the adversary node behavior and survey the usable attack detection techniques.**

After pointing out statically configured countermeasure methods efficiency concerns, a new approach was proposed to overcome these issues by *Chapter 4*. To gain this objective, the attack detection methods were surveyed. However, the lack of side-channel timing attack detection methods in a bibliographic review led to similar attack detection methods. To achieve usable detection methods (mainly cache hit ratio and hop counts), similar cache poisoning detection methods were surveyed and pointed can be adapted for the side-channel timing attack. Through the detection methods, the adversary face can be detected to apply a countermeasure mechanism.

**iv. Develop an attack model to increase the success of the attack for multiple targets.**

The traditional side-channel timing attack may be configurable for a single target which may affect the performance of the attack.

In this work, an attack model is designed to improve the attack success for multiple targets. To achieve that, this attack model is inspired by brute-force design which is considered a recent model and commonly used in today's attack modeling. Through the brute-force attack based implementation, the adversary can obtain multiple targets as designed in *Chapter 4*.

Additionally, the brute-force is designed based on the random function to attack targets randomly instead of sequential order. Through this design, the adversary may also identify the scope of the attack on different NDN applications. For instance, an adversary may target the certificate on trusted applications such as VoNDN.

**v. Design and propose a privacy model to preserve content privacy while also not compromising the NDN distribution performance.**

Depending on the attack configuration and application, the scope of the attack can be different. The adversary may obtain the consumer and producer locations through the attack. Therefore, in this work, multiple detection methods are proposed to be used on privacy-sensitive applications such as VoNDN.

On the other hand, the detection methods were used to obtain the severity of the attack. Through this identification, the attack is classified by minor, moderate, and severe to apply multiple countermeasures methods. There multiple detection and countermeasure based privacy model "detection and defense (DaD)" was presented in *Chapter 4*. Through the DaD algorithm, the adversary face can be detected to apply multiple countermeasures depending on the attack severity.

**vi. The attack and DaD implementations are based on NDN applications.**

To achieve the work objectives, the experimental frameworks and scenario implementations were addressed in *Chapter 5*. In this work, the scenario was implemented and simulated on the NDN simulator. Also, used ndnSIM and its forwarding daemon (NFD) components were evaluated by bibliographic and developer guidelines in this *Chapter 5*.

The implemented scenario results were presented in *Chapter 6*. The NDNtube and VoNDN applications were developed to simulate the brute-force findings. To achieve realistic results, the attack scenario was developed and applied to network topologies such as AT&T and NDN-testbed. The adversary nodes were configured to target the name of the content and certificate. The results also analyzed several attack scenarios on each application. Also, the statically configured countermeasure and dynamically configured DaD implementations result were analyzed. The results showed that the DaD was able to detect the adversarial nodes to applied the countermeasures to only the detected faces.

## 7.3 Summary of Main Contributions

Regarding scenario findings and results, the achieved main contributions of the thesis was summarized by the following items:

**i. Propose a privacy model approach (DaD) to preserve content privacy.**

The countermeasure efficiency issues were pointed out by other research works. To overcome this issue, the detection methods were proposed by [88], [87], and [89] to distinguish the legitimate nodes from the adversaries.

The possible side-channel timing attack and an attack detection approaches were proposed by [88]. The detection was identified by the cache hit ratio that can be used to identify the adversary node to also apply additional delay methods (fixed, random, and unpredictable) to mitigate the attack.

The traditional attack model and detection results were presented by the work [87]. In addition to the cache hit ratio detection metric, the CRT values were proposed and analyzed to detect an adversary on streaming application (16 nodes tree-topology). It was shown that the CRT values were decreased between 37.80% and 53.64% in under attack compared with no-attack periods to detect an adversary [87]. Also, an algorithm was presented to mitigate the attack that was based on CRT and cache hit ratio threshold values.

The sophisticated detection based privacy model was presented by [89] which was called Detection and Defense (DaD). It is based on multiple detections and countermeasures methods to meet with the "*perfect privacy*" approach. In DaD, the detection methods used to detect the adversary face to distinguish from the legitimate nodes. Also, the severity of the attack can be obtained by periodically attack detected faces. Thus, attack detection can be also used to apply multiple countermeasures methods to mitigate the attack by also obtaining their severity.

**ii. Design an attack model that can be used to increase the multiple targets.**

The single segment based attack was implemented by the work [87]. The attack scenario based on tree topology and adversaries has targeted the video-like segments to success in the attack. The success of the attack was calculated up to 90% based on the cache hit ratio on targeted routers.

To meet with today's attack models, the single segment based attack was upgraded by brute-force implementation as presented by the [89]. The adversary was able to define multiple targets to increase the success of the attack. Additionally, a randomized function was implemented to brute-force design to maximize the attack success. To achieve realistic results and adversary behavior, the attack scenario was implemented on the AT&T topology.

**iii. Implement the brute-force attack and DaD algorithm.**

The brute-force attack was implemented on AT&T and NDN-testbed topologies to obtain realistic results by [89] and [98] work. It was shown that the cache hit ratio was obtained $\approx$20% for the attack routers on the VoNDN-like scenario. Through this attack, the adversaries able to obtain the $\approx$65% of video-segment locations. In NDNtube, the adversaries were able to locate the targets as following the clusters: *i.* 30.4% for first, *ii.* 18.2% for neighbor, and *iii.* 51.4% for away routers (*Subsection 6.3.1*).

On the other hand, the work [98] implemented the brute-force attack on a trusted VoNDN-like application scenario that was developed and simulated on NDN-testbed topology. An attack model called brute-force was that increases the success of attack compared to traditional attack designs. The brute-force configured adversaries have targeted 252 certificates these are cached by NDN-testbed edge router. The success of attacks obtained by following: 0.9% CHR in best- route, 50.4% CHR in multicast communications. Also, 35% of adversaries were able to determine the location of cached targets by following: 70.8% cached by the closest, 17.4% cached by the neighbor, and 11.8% cached by away routers.

**iv. Analyze the DaD mitigation results on trust-based VoNDN-like scenario.**

The DaD was implemented as a cache policy to mitigate the brute-force attack on VoNDN as presented by [98]. The detection was based on the cache hit ratio with threshold values. In this scenario, the static countermeasure method (probabilistic caching) was applied and dynamic DaD results were presented and compared. The results showed that the DaD improved the attack mitigation by about 39.1% in best-route and 36.6% in multicast while protecting the legitimate requests when compared to the traditional static probabilistic configurations.

To illustrate that, the DaD can be used to preserve the legitimate request while mitigating the attack, the mean of legitimate hop counts were analyzed during the attack period. The results showed that the DaD (average ≈2.38) preserved ≈2.1% legitimate request, compared to statically configured probabilistic countermeasure method (average ≈2.43).

**v. Analyze the DaD mitigation results on NDNtube-like scenario.**

In default scenario (no-countermeasure applied), the average CHR is obtained ≈16.4% under nfd:LRU policy (*Subsection 6.3.1*). To mitigate the attacks, the countermeasures are attempting to decrease the CHR which is obtained under no-countermeasure applied nfd:LRU (default) scenario.

The DaD dynamically detected the attack and took countermeasure actions instead of statically configured routers to mitigate the attack [journal-2 [1]]. It was shown that the DaD was dynamically mitigated the attack in NDNtube and CHR was obtained 0.7% in the attack period. This result also illustrated that all DaD countermeasures were applied to adversarial faces because the attack was considered as severe.

Also, the statically applied countermeasure results were presented and compared with the scenario of the DaD. The CHR results were obtained by the following: *i.* 4.1% in nfd:probabilistic and *ii.* 3.7% in nfd:freshness respectively.

The countermeasure results showed that the CHR is decreasing under the countermeasure methods compared to the no-countermeasure(s) applied scenario. These countermeasures results are showed that the attacks are more mitigated in DaD (≈0.7%) configuration compared to statically configured countermeasures (probabilistic ≈4.1% and freshness ≈3.7%) in NDNtube (*Subsection 6.3.3*).

## 7.4 Future Works

This section conceptually suggests possible future works these can be the next step or alternative work for this research.

- The attack scope and findings can be different on applications. Thus, the brute-force attack scenario can be implemented on different NDN applications to study its findings. For instance, ideally, the vehicular location can be obtained through the scenario that we developed on the VANET NDN

[1] journal-2 is under-preparation (Section 1.4)

application [99]. To mitigate this an optimal DaD can be proposed. The scenario findings and DaD design can be studied by further works.

- In this work, the scenarios were simulated on ndnSIM. It was used to collect and simulate realistic application simulations on the real-set topologies. To achieve the adversary results, the NDN applications (NDNtube and VoNDN) can be implemented over the real NDN-testbed by possible future implementations. However, currently, this can be still a challenge considering access to nodes and NDN-testbed policies.

**Limitations of work**. In this work, the detection thresholds are suggested based on ndnSIM simulation experiences and application behaviors (NDNtube and VoNDN). In both applications, the attacks are detected dynamically based on pre-defined detection CHR threshold values. However, these detection threshold computations can be defined by further tuning or automatic process using machine learning algorithms according to traffic patterns and complex networks.

## 7.5 Final Considerations

This work presented a new approach to mitigating the side-channel timing attack on NDN content privacy. The proposed countermeasure methods by researches may be considered to be against to NDN paradigm. To overcome this issue, this work proposed detection methods to distinguish the adversary from legitimate nodes. The results showed that the countermeasures efficiency concerns can be overcome by identifying the adversary.

In this work, some applications were re-produced because of the lack of accessible implementation files. Therefore, the applications (except the main contribution) files are publicly available to support future cache privacy-related projects in NDN.

On the other hand, the ndnSIM and NFD can be stated as limited for cache-related developments. The ndnSIM and NFD developers can be more focused on cache-related works to enable future developments, considering the side-channel timing attack is a threat to content privacy in NDN.

# BIBLIOGRAPHY

[1] K. Stamos, G. Pallis, and A. Vakali, "Caching techniques on CDN simulated frameworks," *Lecture Notes in Electrical Engineering*, vol. 9 LNEE, pp. 127–153, 2008. [Online]. Available: https://doi.org/10.1007/978-3-540-77887-5_5

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies - CoNEXT '09*, vol. 30, no. 2. New York, New York, USA: ACM Press, 2009, p. 1. [Online]. Available: https://doi.org/10.1145/1658939.1658941

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, jul 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656887

[4] Y. Yu, A. Afanasyev, D. Clark, K. Claffy, V. Jacobson, and L. Zhang, "Schematizing Trust in Named Data Networking," in *Proceedings of the 2nd International Conference on Information-Centric Networking - ICN '15*, vol. 0030. New York, New York, USA: ACM Press, 2015, pp. 177–186. [Online]. Available: https://doi.org/10.1145/2810156.2810170

[5] Y. Yu, "Usable Security For Named Data Networking," Ph.D. dissertation, University of California, 2016. [Online]. Available: https://pdfs.semanticscholar.org/2ab3/65161a0d1703b65072e3bcd3f28d467e1c71.pdf

[6] D. Kulinski and J. Burke, "NDNVideo : Random-access Live and Pre-recorded streaming using NDN," *Technical Report NDN-0007*, no. September, pp. 1–17, 2012. [Online]. Available: http://www.named-data.net/techreport/TR007-streaming.pdf

[7] D. Van Jacobson, M. Stewart, J. Thornton, and R. Braynard, "VoCCN: Voice Over Content-Centric Networks," *ReArch*, 2009. [Online]. Available: https://doi.org/10.1145/1658978.1658980

[8] G. Zhang, S. Fischer-Huebner, L. A. Martucci, and S. Ehlert, "Revealing the Calling History of SIP VoIP Systems by Timing Attacks," in *2009 International Conference on Availability, Reliability and Security*. IEEE, 2009, pp. 135–142. [Online]. Available: https://doi.org/10.1109/ARES.2009.129

[9] N. Kamoltham, K. N. Nakorn, and K. Rojviboonchai, “From NS-2 to NS-3 - Implementation and evaluation,” *2012 Computing, Communications and Applications Conference, ComComAp 2012*, pp. 35–40, 2012.

[10] A. Afanasyev, I. Moiseenko, and L. Zhang, “ndnSIM: NDN simulator for NS-3,” *NDN Technical Report*, pp. 1–7, 2012. [Online]. Available: http://named-data.net/techreport/TR005-ndnsim.pdf

[11] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, “ndnSIM 2 . 0 : A new version of the NDN simulator for NS-3,” NDN, Technical Report NDN-0028, Tech. Rep., 2015. [Online]. Available: https://named-data.net/wp-content/uploads/2013/07/ndn-0028-1-ndnsim-v2.pdf

[12] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. v. Zhang, “ndnSIM 2: An updated NDN simulator for NS-3,” no. NDN-0028, Revision 2, pp. 1–8, 2016. [Online]. Available: https://named-data.net/wp-content/uploads/2016/11/ndn-0028-2-ndnsim-v2.pdf

[13] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. Dibenedetto, C. Fan, D. Pesavento, G. Grassi, G. Pau, H. Zhang, T. Song, H. B. Abraham, P. Crowley, S. O. Amin, V. Lehman, and L. Wang, “NFD Developer’s Guide,” *NDN, Technical Report NDN-0021*, vol. 4, pp. 1–56, 2015. [Online]. Available: https://named-data.net/wp-content/uploads/2016/10/ndn-0021-7-nfd-developer-guide.pdf

[14] Named-data.net, “NDN Testbed - Named Data Networking (NDN),” 2015. [Online]. Available: https://named-data.net/ndn-testbed/

[15] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, “Named Data Networking: A survey,” *Computer Science Review*, vol. 19, pp. 15–55, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.cosrev.2016.01.001

[16] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, L. Wang, P. Crowley, and E. Yeh, “Named Data Networking (NDN) Project,” NDN, Technical Report NDN-0001, Tech. Rep., 2010. [Online]. Available: http://named-data.net/techreport/TR001ndn-proj.pdf

[17] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik, “Cache privacy in named-data networking,” *Proceedings - International Conference on Distributed Computing Systems*, pp. 41–51, 2013. [Online]. Available: https://doi.org/10.1109/ICDCS.2013.12

[18] A. Mohaisen, H. Mekky, X. Zhang, H. Xie, and Y. Kim, “Timing Attacks on Access Privacy in Information Centric Networks and Countermeasures,” *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 6, pp. 675–687, 2015. [Online]. Available: https://doi.org/10.1109/TDSC.2014.2382592

[19] C. Bernardini, S. Marchal, M. R. Asghar, and B. Crispo, "PrivICN: Privacy-preserving content retrieval in information-centric networking," *Computer Networks*, vol. 149, pp. 13–28, 2019. [Online]. Available: https://doi.org/10.1016/j.comnet.2018.11.012

[20] E. W. Felten and M. A. Schneider, "Timing attacks on Web privacy," *Proceedings of the 7th ACM conference on Computer and communications security - CCS '00*, pp. 25–32, 2000. [Online]. Available: https://doi.org/10.1145/352600.352606

[21] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun, "ANDaNA: Anonymous Named Data Networking Application," in *Proceedings of the Network and Distributed System Security Symposium*, 2011. [Online]. Available: http://arxiv.org/abs/1112.2205

[22] S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch, "Information-Centric Networking (ICN) Research Challenges," Tech. Rep., jul 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7927

[23] W. M. , Mosko I., Solis C., "Content-Centric Networking (CCNx) Semantics." [Online]. Available: https://tools.ietf.org/pdf/rfc8569.pdf

[24] L. Peterson, B. Davie, and E. R. van Brandenburg, "Framework for Content Distribution Network Interconnection (CDNI)," Tech. Rep., 2014. [Online]. Available: https://tools.ietf.org/html/rfc7336

[25] T. Dalgleish, J. M. G. Williams, A.-M. J. Golden, N. Perkins, L. F. Barrett, P. J. Barnard, C. Au Yeung, V. Murphy, R. Elward, K. Tchanturia, and E. Watkins, *Content Delivery Networks*, ser. Lecture Notes Electrical Engineering, R. Buyya, M. Pathan, and A. Vakali, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 9, no. 1. [Online]. Available: https://doi.org/10.1007/978-3-540-77887-5%0A

[26] M. Pathan, R. Buyya, and A. Vakali, "Content delivery networks: State of the art, insights, and imperatives," *Lecture Notes in Electrical Engineering*, vol. 9 LNEE, pp. 3–32, 2008. [Online]. Available: https://doi.org/10.1007/978-3-540-77887-5_1

[27] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, "Naming in content-oriented architectures," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking - ICN '11*. New York, New York, USA: ACM Press, 2011, p. 1. [Online]. Available: https://doi.org/10.1145/2018584.2018586

[28] W. You, B. Mathieu, and G. Simon, "How to make content-centric networks interwork with CDN networks," in *2013 Fourth International Conference on the Network of the Future (NoF)*. IEEE, oct 2013, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NOF.2013.6724511

[29] C. Ghali, G. Tsudik, and C. A. Wood, “When Encryption is Not Enough : Privacy Attacks in Content-Centric Networking,” *Proceedings of ACM ICN*, 2017. [Online]. Available: https://doi.org/10.1145/3125719.3125723

[30] “Information-Centric Networking Research Group.” [Online]. Available: https://irtf.org/icnrg

[31] P. Jacobson, V. and Burke, J. and Zhang, L. and Zhang, B. and Claffy, K. and Papadopoulos, C. and Abdelzaher, T. and Wang, L. and Halderman, J. and Crowley, “Named Data Networking Next Phase (NDN-NP) Project May 2014 - April 2015 Annual Report,” Named Data Networking (NDN), Tech. Rep., 2015. [Online]. Available: https://named-data.net/project/annual-progress-summaries/ndn-ar2017/

[32] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, “Adaptive forwarding in named data networking,” *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 3, p. 62, 2012. [Online]. Available: http://dx.doi.org/10.1145/2317307.2317319

[33] B. Zhang, L. Zhang, I. Moiseenko, A. Afanasyev, J. Shi, Y. Yu, W. Shang, Y. Li, S. Mastorakis, Y. Huang, J. P. Abraham, E. Newberry, T. Liang, K. Schneider, S. Dibenedetto, C. Fan, S. Shannigrahi, C. Papadopoulos, D. Pesavento, G. Grassi, G. Pau, H. Zhang, T. Song, H. Yuan, H. B. Abraham, P. Crowley, S. O. Amin, V. Lehman, M. Chowdhury, A. Gawande, L. Wang, and N. Gordon, “NFD Developer’s Guide,” *NDN, Technical Report NDN-0021*, no. June, 2018. [Online]. Available: https://named-data.net/wp-content/uploads/2016/10/ndn-0021-7-nfd-developer-guide.pdf

[34] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. Dibenedetto, C. Fan, D. Pesavento, G. Grassi, G. Pau, H. Zhang, T. Song, H. B. Abraham, P. Crowley, S. O. Amin, V. Lehman, and L. Wang, “NFD Developer ’ s Guide,” NDN, Technical Report NDN-0021, Tech. Rep., 2015. [Online]. Available: https://named-data.net/wp-content/uploads/2016/10/ndn-0021-7-nfd-developer-guide.pdf

[35] C. Ghali, G. Tsudik, and E. Uzun, “Elements of Trust in Named-Data Networking,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 12–19, feb 2014. [Online]. Available: http://arxiv.org/abs/1402.3332

[36] Y. Yu, “Public Key Management in Named Data Networking,” *NDN, Technical Report NDN-0029*, 2015. [Online]. Available: http://named-data.net/publications/techreports/ndn-0029-1-public-key-management-ndn/

[37] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” Tech. Rep. 1, 2008. [Online]. Available: https://tools.ietf.org/html/rfc5280

[38] C. Bian, Z. Zhu, A. Afanasyev, E. Uzun, and L. Zhang, "Deploying key management on NDN testbed," *Ndn, Tr*, vol. 9, no. February, 2013. [Online]. Available: https://named-data.net/wp-content/uploads/TRpublishkey-rev2.pdf

[39] D. K. Smetters and V. Jacobson, "Securing network content," *named-data.net*, 2009. [Online]. Available: https://named-data.net/wp-content/uploads/securing-network-content-tr.pdf

[40] L. Wang, V. Lehman, A. K. Mahmudul Hoque, B. Zhang, Y. Yu, and L. Zhang, "A Secure Link State Routing Protocol for NDN," *IEEE Access*, vol. 6, pp. 10 470–10 482, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2789330

[41] "NDN Regular Expression — ndn-cxx: NDN C++ library with eXperimental eXtensions 0.5.1-74-gb1a2a4b4 documentation." [Online]. Available: https://named-data.net/doc/ndn-cxx/current/

[42] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang, "Securing building management systems using named data networking," *IEEE Network*, vol. 28, no. 3, pp. 50–56, 2014. [Online]. Available: https://doi.org/10.1109/MNET.2014.6843232

[43] P. Gusev, Z. Wang, J. Burke, L. Zhang, T. Yoneda, R. Ohnishi, and E. Muramoto, "Real-time streaming data delivery over Named Data Networking," *IEICE Transactions on Communications*, vol. E99B, no. 5, pp. 974–991, 2016. [Online]. Available: https://doi.org/10.1587/transcom.2015AMI0002

[44] S. Mastorakis, P. Gusev, and A. Afanasyev, "Real-Time Data Retrieval in Named Data Networking," in *IEEE International Conference on Hot Information-Centric Networking*. IEEE, 2018. [Online]. Available: https://named-data.net/publications/hoticn18realtime-retrieval/

[45] Z. Zhu, A. Afanasyev, Y. Yu, and L. Zhang, "ChronoChat : a Server-less Multi-User Instant Message Application Over NDN," NDN, Technical Report NDN-0008, Tech. Rep., 2014. [Online]. Available: http://named-data.net/techreport/TR008-chronos.pdf

[46] S. Signorello, M. R. Palattella, and L. A. Grieco, "Security challenges in future NDN-enabled VANETs," *Proceedings - 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 10th IEEE International Conference on Big Data Science and Engineering and 14th IEEE International Symposium on Parallel and Distributed Proce*, pp. 1771–1775, 2016. [Online]. Available: https://doi.org/10.1109/TrustCom.2016.0272

[47] "Repo Protocol Specification - repo-ng - NDN project issue tracking system." [Online]. Available: https://redmine.named-data.net/projects/repo-ng/wiki/Repo_Protocol_Specification

[48] M. Ambrosin, A. Compagno, M. Conti, C. Ghali, and G. v. Tsudik, "Security and Privacy Analysis of NSF Future Internet Architectures," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 2, pp. 1418–1442, 2018. [Online]. Available: https://doi.org/10.1109/COMST.2018.2798280

[49] M. Ambrosin, A. Compagno, M. Conti, C. Ghali, and G. Tsudik, "Security and Privacy Analysis of National Science Foundation Future Internet Architectures," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 1418–1442, 2018. [Online]. Available: https://doi.org/10.1109/COMST.2018.2798280

[50] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, Privacy, and Access Control in Information-Centric Networking: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 1, pp. 556–600, 2018. [Online]. Available: https://doi.org/10.1109/COMST.2017.2749508

[51] W. Ding, Z. Yan, and R. H. Deng, "A Survey on Future Internet Security Architectures," *IEEE Access*, vol. 4, pp. 4374–4393, 2016. [Online]. Available: https://doi.org/10.1109/ACCESS.2016.2596705

[52] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014. [Online]. Available: https://doi.org/10.1109/SURV.2013.070813.00063

[53] A. Chaabane, E. De Cristofaro, M.-A. Kaafar, and E. Uzun, "Privacy in Content-Oriented Networking: Threats and Countermeasures," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 26–33, 2012. [Online]. Available: https://doi.org/10.1145/2500098.2500102

[54] A. Afanasyev, X. Jiang, Y. Yu, J. Tan, Y. Xia, A. Mankin, and L. Zhang, "NDNS: A DNS-like name service for NDN," *2017 26th International Conference on Computer Communications and Networks, ICCCN 2017*, no. Section II, 2017. [Online]. Available: https://doi.org/10.1109/ICCCN.2017.8038461

[55] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *National Renewable Energy Laboratory (NREL)*, 1996, pp. 104–113. [Online]. Available: https://doi.org/10.1007/3-540-68697-5_9

[56] Tobias Lauinger, "Security & Scalability of Content-Centric Networking," no. September, p. 60, 2010. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/2275/1/ccn-thesis.pdf

[57] T. Lauinger, N. Laoutaris, and P. Rodriguez, "Privacy Risks in Named Data Networking: What is the Cost of Performance?" *Acm Sigcomm*, vol. 42, no. 5, pp. 54–57, 2012. [Online]. Available: https://doi.org/10.1145/2378956.2378966

[58] A. Compagno, M. Conti, E. Losiouk, G. Tsudik, and S. Valle, "A Proactive Cache Privacy Attack on NDN," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, apr 2020, pp. 1–7. [Online]. Available: https://doi.org/10.1109/NOMS47738.2020.9110318

[59] Douglas E. Comer, *Internetworking with TCP/IP*. Prentice-Hall, 2000. [Online]. Available: https://dl.acm.org/citation.cfm?id=518740

[60] P. Gusev and J. Burke, "Ndn-Rtc," *Proceedings of the 2nd International Conference on Information-Centric Networking - ICN '15*, pp. 117–126, 2015. [Online]. Available: https://doi.org/10.1145/2810156.2810176

[61] L. Wang, "NDNlive and NDNtube : Live and Prerecorded Video Streaming over NDN," *NDN Technical Report NDN-0031*, pp. 1–10, 2015. [Online]. Available: https://named-data.net/publications/techreports/ndn-0031-1-ndnlive-ndntube/

[62] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "ACT: Audio Conference Tool Over Named Data Networking," *ACM SIGCOMM workshop on Information-centric networking*, vol. 11, p. 68, 2011. [Online]. Available: https://doi.org/10.1145/2018584.2018601

[63] M. Handley, V. Jacobson, and C. Perkins, "SDP: Session Description Protocol," Network Working Group, Tech. Rep., 2006. [Online]. Available: https://doi.org/10.17487/rfc4566

[64] R. Birke, M. Mellia, M. Petracca, and D. Rossi, "Experiences of VoIP traffic monitoring in a commercial ISP," *International Journal of Network Management*, vol. 20, no. 5, pp. 339–359, aug 2010. [Online]. Available: https://doi.org/10.1002/nem.758

[65] T. Lauinger, N. Laoutaris, and P. Rodriguez, "Privacy Implications of Ubiquitous Caching in Named Data Networking Architectures," *Acm Sigcomm*, vol. 42, no. 5, pp. 54–57, 2012. [Online]. Available: https://old.iseclab.org/papers/ccn-cache-attacks-iseclab-0812-001.pdf

[66] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, vol. 179, no. 7. IEEE, may 2008, pp. 35–49. [Online]. Available: https://doi.org/10.1109/SP.2008.21

[67] S. Chen, R. Wang, X. F. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 191–206, 2010. [Online]. Available: https://doi.org/10.1109/SP.2010.20

[68] J. Peterson and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)," The Internet Society (2006), Tech. Rep., 2006. [Online]. Available: https://tools.ietf.org/pdf/rfc4474.pdf

[69] S. Schinzel, "An Efficient Mitigation Method for Timing Side Channels on the Web," *2nd International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*,

pp. 1–6, 2011. [Online]. Available: https://www.researchgate.net/publication/336209882_An_Efficient_Mitigation_Method_for_Timing_Side_Channels_on_the_Web

[70] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking - ICN '12*. New York, New York, USA: ACM Press, 2012, p. 55. [Online]. Available: https://doi.org/10.1145/2342488.2342501

[71] D. Chaum and E. Van Heyst, "Group signatures," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 547 LNCS, no. iii, pp. 257–265, 1991. [Online]. Available: https://doi.org/10.1007/3-540-46416-6_22

[72] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signatures," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3152, pp. 41–55, 2004.

[73] R. Wiangsripanawan, W. Susilo, and R. Safavi-Naini, "Design principles for low latency anonymous network systems secure against timing attacks," *Conferences in Research and Practice in Information Technology Series*, vol. 68, pp. 183–191, 2007. [Online]. Available: https://dl.acm.org/doi/10.5555/1274531.1274553

[74] D. Kondo, T. Silverston, V. Vassiliades, H. Tode, and T. Asami, "Name filter: a countermeasure against information leakage attacks in named data networking," *IEEE Access*, vol. 6, pp. 65 151–65 170, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2877792

[75] A. Compagno, M. Conti, P. Gasti, L. V. Mancini, and G. Tsudik, "Violating consumer anonymity: Geo-locating nodes in named data networking," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9092, pp. 243–262, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-28166-7_12

[76] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. M. Maggs, K. C. Ng, V. Sekar, and S. Shenker, "Less Pain, Most of the Gain: Incrementally Deployable ICN," *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, vol. 43, no. 4, p. 147, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486001.2486023

[77] R. Gorrieri, R. Lanotte, A. Maggiolo-Schettini, F. Martinelli, S. Tini, and E. Tronci, "Automated analysis of timed security: a case study on web privacy," *International Journal of Information Security*, vol. 2, no. 3-4, pp. 168–186, 2004. [Online]. Available: http://link.springer.com/10.1007/s10207-004-0037-9

[78] N. Kumar, A. K. Singh, and A. Aleem, "Security Attacks in Named Data Networking : A Review and Research Directions," vol. 34, no. 6, pp. 1319–1350, 2019. [Online]. Available: https://doi.org/10.1007/s11390-019-1978-9

[79] X. Wang, "Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet Categories and Subject Descriptors," in *ACM conference on Computer and communications security*. ACM Press, 2005, pp. 81–91. [Online]. Available: https://doi.org/10.1145/1102120.1102133

[80] T. S. Saponas, J. Lester, and C. Hartung, "Devices That Tell On You : Privacy Trends in Consumer Ubiquitous Computing," *16th USENIX Security Sympsosium 2007*, pp. 1–23, 2007. [Online]. Available: https://homes.cs.washington.edu/~sagarwal/tracker.pdf

[81] A. Bortz and D. Boneh, "Exposing private information by timing web applications," *Proceedings of the 16th international conference on World Wide Web - WWW '07*, p. 621, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1242572.1242656

[82] T. N. D. Pham, C. K. Yeo, N. Yanai, and T. Fujiwara, "Detecting flooding attack and accommodating burst traffic in delay-tolerant networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 795–808, 2018. [Online]. Available: https://doi.org/10.1109/TVT.2017.2748345

[83] A. Hayes, "Network service authentication timing attacks," *IEEE Security and Privacy*, vol. 11, no. 2, pp. 80–82, 2013.

[84] J. Owens and J. Matthews, "A Study of Passwords and Methods Used in Brute-Force SSH Attacks," *leet*, 2008. [Online]. Available: https://people.clarkson.edu/~owensjp/pubs/leet08.pdf

[85] L. Yao, Z. Fan, J. Deng, X. Fan, and G. Wu, "Detection and Defense of Cache Pollution Attacks Using Clustering in Named Data Networks," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. c, pp. 1–1, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8493281/

[86] M. Conti, P. Gasti, and M. Teoli, "A lightweight mechanism for detection of cache pollution attacks in Named Data Networking," *Computer Networks*, vol. 57, no. 16, pp. 3178–3191, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2013.07.034

[87] E. Dogruluk, A. Costa, and J. Macedo, "Identifying Previously Requested Content by Side-Channel Timing Attack in NDN," in *Communications in Computer and Information Science*, aug 2018, vol. 878, pp. 33–46. [Online]. Available: https://doi.org/10.1007/978-3-319-94421-0_3

[88] E. Dogruluk, A. Costa, and J. Macedo, "Evaluating privacy attacks in Named Data Network," in *Proceedings - IEEE Symposium on Computers and Communications*, vol. 2016-Augus. IEEE, jun 2016, pp. 1251–1256. [Online]. Available: https://doi.org/10.1109/ISCC.2016.7543908

[89] E. Dogruluk, A. Costa, and J. Macedo, "A Detection and Defense Approach for Content Privacy in Named Data Network," *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, 2019. [Online]. Available: https://doi.org/10.1109/NTMS.2019.8763835

[90] L. Deng, Y. Gao, Y. Chen, and A. Kuzmanovic, "Pollution attacks and defenses for Internet caching systems," *Computer Networks*, vol. 52, no. 5, pp. 935–956, 2008. [Online]. Available: https://doi.org/10.1016/j.comnet.2007.11.019

[91] "ns-3 Tutorial," Tech. Rep., 2017. [Online]. Available: https://www.nsnam.org/docs/release/3.18/tutorial/ns-3-tutorial.pdf

[92] NS3 Development Team, "NS3 discrete-event network simulator for Internet systems," 2020. [Online]. Available: https://www.nsnam.org/

[93] Named-data.net, "ndn-cxx overview — ndn-cxx: NDN C++ library with eXperimental eXtensions 0.4.0-beta2-10-g664dc03 documentation," 2015. [Online]. Available: https://named-data.net/doc/ndn-cxx/current/

[94] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies With Rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2–16, feb 2004. [Online]. Available: http://ieeexplore.ieee.org/document/1268075/

[95] N. Spring, R. Mahajan, and T. Anderson, "Quantifying the Causes of Path Inflation," p. 113, 2003. [Online]. Available: https://doi.org/10.1145/863969.863970

[96] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," p. 231, 2002. [Online]. Available: https://doi.org/10.1145/637235.637237

[97] J. Dehart, "Status and Upcoming Changes NDN Retreat," 2016. [Online]. Available: https://named-data.net/wp-content/uploads/2016/11/ndn1611_jdehart.pdf

[98] E. Dogruluk, O. Gama, A. D. Costa, and J. Macedo, "Public Key Certificate Privacy in VoNDN: Voice Over Named Data Networks," *IEEE Access*, vol. 8, pp. 145 803–145 823, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3014898

[99] G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang, "VANET via Named Data Networking," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, vol. 19, no. 8. IEEE, apr 2014, pp. 410–415. [Online]. Available: http://ieeexplore.ieee.org/document/6849267/