Security Certification Scheme for Content-centric Networks

Marco Anisetti, Claudio A. Ardagna, Filippo Berto, Ernesto Damiani
Dipartimento di Informatica
Universita' degli studi di Milano
Milano, Italy
{firstname.lastname}@unimi.it

***Abstract*—Content-centric networking is emerging as a credible alternative to host-centric networking, especially in scenarios of large-scale content distribution and where privacy requirements are crucial. Recently, research on content-centric networking has focused on security aspects and proposed solutions aimed to protect the network from attacks targeting the content delivery protocols. Content-centric networks are based on the strong assumption of being able to access genuine content from genuine nodes, which is however unrealistic and could open the door to disruptive attacks. Network node misbehavior, either due to poisoning attacks or malfunctioning, can act as a persistent threat that goes unnoticed and causes dangerous consequences. In this paper, we propose a novel certification methodology for content-centric networks that improves transparency and increases trustworthiness of the network and its nodes. The proposed approach builds on behavioral analysis and implements a continuous certification process that collects evidence from the network nodes and verifies their non-functional properties using a rule-based inference model. Utility, performance, and soundness of our approach have been experimentally evaluated on a simulated Named Data Networking (NDN) network targeting properties availability, integrity, and non-repudiation.**

***Index Terms*—Content-centric networking; named data networking; security; certification;**

I. Introduction

Today, the interest in content-centric networking as a substitute for the common TCP/IP network stack is gradually increasing [1]–[5]. This is especially true in scenarios where in-protocol content distribution and privacy features are of paramount importance [6]. The research and development community has made great strides in the implementation of the content-centric paradigm, concentrating its efforts on functional aspects and performance [6], [7]. Research on security aspects instead has mostly focused on specific attacks [8], [9] and countermeasures [10]–[13] missing the big picture. Transparency and trustworthiness of content-centric networks are in fact a major hurdle against its widespread adoption and can open the door to persistent threats that affect the network behavior to its foundation. In addition, weaknesses to poisoning attacks and system malfunctioning can impair the entire network operation [14], [15].

In this paper we present a certification methodology for content-centric networks continuously certifying non-functional properties of network nodes in operation. Our methodology is based on a set of inference rules mapping networking behavior to non-functional properties. Inference rules are evaluated according to measurements on the status of the network collected from its components. Being able to certify non-functional properties of the network can support an effective QoS approach, where network functioning is adapted to evolving conditions, increasing network trustworthiness and quality. Our network-level certification approach complements modern composite applications based on microservices, paving the way to a new generation of certified compositions tightly intertwined with networking technologies [4], [5], [16], [17]. It also increases the attractiveness of content-centric networking for ISPs or cloud providers interested in offering certified services.

While our methodology is general enough to cope with any content-centric network implementation, we focus on Named Data Networking (NDN) [2]. We then extend NDN with our certification methodology towards certification-aware NDN architecture and networking services. As an example of its utility, let us consider a financial scenario where the integrity of the transmitted data (e.g., bank transactions) is a critical requirement. NDN protocol can protect data integrity by requiring each content to be signed by its producer. The service application can then rely on this feature, and the consumer can verify both the integrity and the origin of each received content. However, its effectiveness strongly depends on the correct behavior of the network nodes (e.g., the producer and consumer nodes use a valid public-private key pair for their communications). Our certification methodology can fill in this gap supporting continuous and automatic verification of network-level properties, preventing data breaches at communication layer.

The contribution of this paper is threefold. We first extend NDN architecture to support network certification maintaining compatibility at protocol layer. This is achieved by extending the reporting capabilities of NDN nodes and adding Certification Agents (CAs). We then propose a novel certification methodology based on inference rules and measurements that awards certificates proving non-functional properties on NDN nodes. We finally present an effective implementation of our methodology within the NDN ecosystem, involving the certification of multiple security properties.

The remaining of this paper is organized as follows. Section II presents our system model. Section III describes

our certification methodology. Section IV presents our the certification process. Section V experimentally evaluates the soundness and performance of our approach in a simulated NDN network. Section VI presents the related work, while Section VII draws our final remarks.

II. System model

Named Data Networking (NDN) is a content-centric network protocol, where consumers and producers communicate to exchange a given content using interest and data packets. Interest packets are sent by a content consumer to request a certain content in the form of a complete name or a prefix. A content producer receives an interest packet and responds with a data packet containing the full content name and the actual data. Every data packet is signed by its producer and has a freshness period that limits its validity in time. NDN nodes can either cache data packets and immediately answer a matching interest request or forward the packet to one or more adjacent nodes. The main advantage of NDN over traditional networks is the protocol inherent capability of implementing a distributed Content Distribution Network (CDN), with each node acting as a cache for locally popular contents. Moreover, both interest and data packets do not contain any direct reference to their consumers or producers, considerably improving the privacy of the users. NDN nodes communicate through the Named Data Networking Forwarding Daemon (NFD), which implements the routing capabilities of the network. NFD uses three main data structures: *i)* Content Store (CS), the cache of the node, storing data packets in memory for further distribution; *ii)* Forwarding Information Base (FIB) holding information about the preferred network interface through which forwarding of interest packets is most effective; *iii)* Pending Interest Table (PIT) keeping record of the forwarded interests for each network interface, preventing the node to flood the rest of the network with repeated requests.

In this paper, we extend the NDN architecture to achieve higher quality and trustworthiness via continuous certification of non-functional properties. Certification, in fact, increases trustworthiness by providing verifiable evidence that a specific property holds for a given system. Figure 1 shows the architecture and interaction flows of our certification-aware NDN. It is composed of three main parties whose communications are denoted with black arrows.

- *CAs* execute the certification process and collect evidence about the status of the NDN network nodes. CAs verify the compliance of the collected evidence to a given non-functional property and, in case of positive evaluation, award a certificate to the NDN node. In case of negative evaluation, an existing certificate can be either updated or revoked. CAs are add-ons, compared to traditional NDN architecture, needed for enabling continuous certification in the operation environment.
- *NDN network nodes* are servers running NFD processes, acting as routers for NDN traffic, and exposing services to the users. These processes are extended to provide CAs with the evidence at the basis our certification process (see Section V-A for details). The network nodes are securely bound with certificates on a predefined prefix, allowing other users of the network to verify their non-functional properties.
- *Users* interact with NDN certified network nodes to execute services on a trusted networking layer.

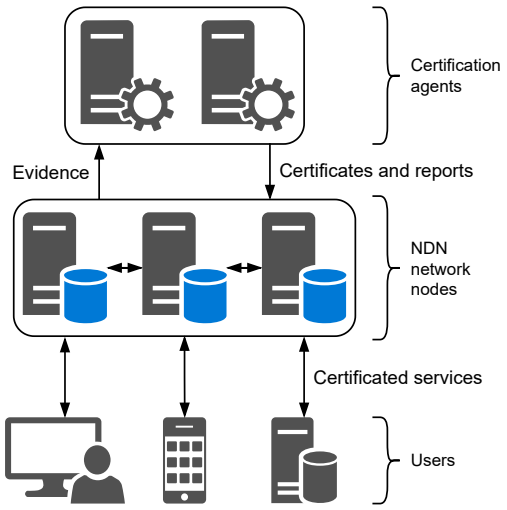


Fig. 1. Certification-aware NDN scenario

We note that the certification process is usually executed by a trusted certification authority with the support of accredited labs. In this paper we adopt the chain of trust described in [18], where the certification authority delegates the certification framework (i.e., CAs) to work as accredited lab in operation. The certification authority remains responsible to model the activities needed to verify a given non-functional property. We remark that our extensions towards a certification-aware NDN are fully compatible with the standard NDN protocol and implementation. We also note that all network communications between the monitored nodes and the CAs are natively implemented, using the NDN protocol.

III. Certification Methodology

Figure 2 shows our certification methodology. The CA executes two tasks to: *i)* collect measurements using computing metrics; *ii)* evaluate inference rules on a target NDN system. Metrics provide measurements on specific aspects of the target NDN network. They capture details on the internal state of a given NDN node in operation. Inference rules are Boolean expressions based on measurements collected by the metrics and model specific run-time behaviors of the system. Measurements, as well as inference rule outcomes, change over time and constitute the evidence of the certification process. Evidence is a fundamental part of our certificate methodology, since it permits to replicate the certification process, thus improving the trust on it. Inference rules can be combined with other inference rules to model more complex behaviors.

In this paper, for simplicity but with no lack of generality, we consider simple Boolean rules. Among all the inference

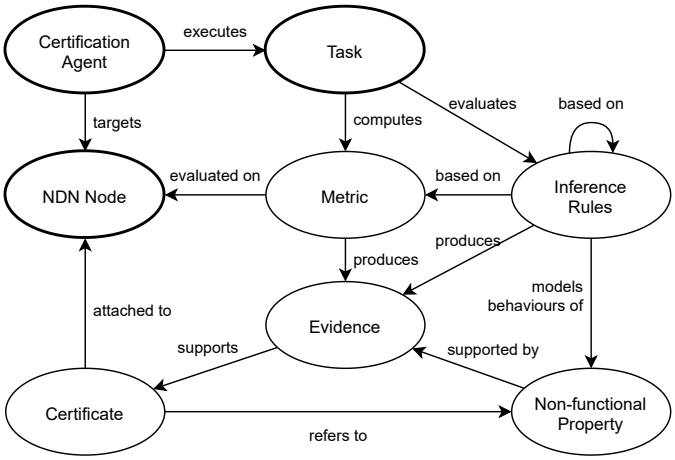


Fig. 2. Certification methodology.

rules, the CA selects the ones needed for supporting the certification of the given non-functional properties. In case the rule evaluation supports the non-functional properties, the certificate is awarded and attached to the corresponding NDN node. Note that, being evaluated on temporal variant rules, the certificate itself has a dynamic life cycle [18] and can be possibly revoked when inference rules are violated in a given time frame.

Below, we detail all components of our certification methodology in Figure 2 (i.e., metrics, inference rules and certificate).

A. Metrics

Metrics are implemented as functions that provide measurements on the status of the NDN system (or a part thereof) at a given time instant. Metrics are designed with the following requirements in mind.

- Metrics must produce a simple yet effective measurement of the system status that can be used to infer non-functional properties.
- The computational effort must be negligible compared to the descriptive value.
- Metrics should represent a minimum set of expressive measures.
- Metrics must show the temporal evolution of the system.
- Metrics must not interfere with the NDN protocol.
- Metrics should be as much as possible scenario and user independent.

Table I presents a list of metrics for NDN nodes. We note that, while a larger number of metrics can provide a more fine-grained and application-tailored representation of the system, a trade-off with the implementation complexity must be considered.

Formally, a metric m is a function of time t, denoted $m(t)$, where t can be either a single time instant or a time window used for the collection of measurements.

Example 1 (Integrity-Related Metrics): Let us consider the NDN packet integrity mechanism used by NDN producers to sign data packets. A default content certificate is chosen from those available as a preset. The public key contained in a content certificate is shared by the node with the network on a predefined prefix, enabling any clients to verify the integrity of the content received by a producer, as well as its identity.

TABLE I
EXAMPLES OF NDN NODE METRICS

Id	Metric	Description
m_1	CS policy name	Which CS policy is enabled
m_2	Maximum size of the CS	How many entries can be stored in the CS
m_3	CS usage	How many entries are stored in the CS
m_4	CS entry statistics	Minimum, maximum, mean and standard deviation of the CS entries memory usage
m_5	Interest forwarding policy	Which forwarding policy is used for each interest packet
m_6	PIT entries	Number of pending interest packets stored in the PIT
m_7	Interest packets size statistics	Minimum, maximum, mean and standard deviation of the incoming interest packets size
m_8	Data packets size statistics	Minimum, maximum, mean and standard deviation of the outgoing data packets size
m_9	Interest packets components statistics	Minimum, maximum, mean and standard deviation of the number of components in the incoming interest packets
m_{10}	Data packets components statistics	Minimum, maximum, mean and standard deviation of the number of components in the outgoing data packets
m_{11}	Contents certificates validity	Time interval of validity of the stored content certificates
m_{12}	Default content certificate	If and which default content certificate is set
m_{13}	Node memory	The total amount of system memory of the NFD node
m_{14}	Packets signature validity statistics	The total amount of valid and invalid signature contents stored in CS

Let us then consider metrics m_{11} and m_{12} in Table I with the scope of measuring the NDN integrity mechanism. The first metric m_{11} focuses on content certificate validity, returning the validity range of each content certificate as follows:

$$m_{11}(t) = \begin{Bmatrix} min := \text{validity start} \\ max := \text{validity end} \end{Bmatrix}$$

where t is the current time instant. The second metric m_{12} focuses on the mechanism configuration, returning a set containing the default certificate if any or an empty set otherwise as follows:

$$m_{12}(t) = \begin{cases} \{c\} & c \text{ is set as default certificate} \\ \emptyset & \text{otherwise} \end{cases}$$

where t is the current time instant.

B. Inference rules

An inference rule is defined according to the following simplified BNF notation.[1]

$$
\begin{aligned}
\langle \text{rule} \rangle &::= \langle \text{simple\_rule} \rangle \mid \langle \text{complex\_rule} \rangle \\
\langle \text{simple\_rule} \rangle &::= \langle \text{v\_expr} \rangle \langle \text{operator} \rangle \langle \text{v\_expr} \rangle \\
\langle \text{complex\_rule} \rangle &::= NOT\ \langle \text{rule} \rangle \mid \langle \text{rule} \rangle \langle \text{bin\_op} \rangle \langle \text{rule} \rangle \\
\langle \text{v\_expr} \rangle &::= \langle \text{m\_val} \rangle \mid \langle \text{v\_expr} \rangle . \langle \text{attribute} \rangle \mid \langle \text{constant} \rangle \mid \\
&\quad \langle \text{m\_transform} \rangle (\langle \text{m\_sequence} \rangle) \mid \\
\langle \text{attribute} \rangle &::= min \mid max \mid avg \mid stdDev \mid \ldots \\
\langle \text{m\_transform} \rangle &::= minimum \mid maximum \mid mean \mid \\
&\quad std\_dev \mid \ldots \\
\langle \text{m\_sequence} \rangle &::= \langle \text{m\_val} \rangle \mid , \langle \text{m\_sequence} \rangle \\
\langle \text{operator} \rangle &::= < \mid \leq \mid = \mid \neq \mid \geq \mid > \\
\langle \text{bin\_op} \rangle &::= \wedge \mid \vee
\end{aligned}
$$

Following this definition, inference rules can be simple rules denoted as $c(t)$ or complex rules denoted as $r(t)$. Complex rules are defined as a Boolean combination of simple and complex rules. Inference rules are designed by the certification authority and used to evaluate behavioral aspects of the system under certification with the aim of proving support for a given non-functional property. Table II shows a set of inference rules. They are built on metrics in Table I and consider δ_t as the evaluation time window expressed in seconds. Although each rule can have a different time window depending on the specific behavioral aspect to be evaluated, for simplicity, we assume a shared time window δ_t.

C. Non-functional property

A non-functional property can be formally defined as follows.

Definition 1: A property $\mathbf{p(t)}$ to be certified is a tuple of the form $<name,\ f(R)>$, where name is taken from a controlled vocabulary like "Confidentiality" and $\mathbf{R}$ is a non-empty set of inference rules used to evaluate the property on the target NDN at time t. $f(R)$ is a Boolean function (typically a conjunction) expressed in terms of rules in $\mathbf{R}$.

We note that a certification process can refer to multiple non-functional properties, and therefore the set of rules to be evaluated is composed of the union of the two sets of rules without repetition.

Example 2 (Integrity-Related Constrained Rules): Let us consider metrics in Example 1 and the inference rules (i.e., simple rules) c_{13} and c_{14} in Table II. c_{13} verifies whether all content certificates stored in the NFD node are valid, that is, the current time is within the minimum and maximum bounds of each content certificate. It can be formally defined as follows:

$$
c_{13}(t) := \begin{cases} \text{True} & \forall\ \text{cert} \in m_{11}(t), \text{cert.min} \leq t \leq \text{cert.max} \\ \text{False} & \text{otherwise} \end{cases}
$$

[1]We omitted trivial definitions for ⟨simple_rule⟩, ⟨complex_rule⟩, ⟨m_val⟩ and ⟨constant⟩.

TABLE II
EXAMPLES OF INFERENCE RULES BASED ON THE METRICS DEFINED IN TABLE I CONSIDERING A TIME WINDOW δ_t AND THE CURRENT TIME INSTANT t EXPRESSED IN SECONDS.

Rule	Expression	Description
$c_1(t)$	$m_1(t) = "lru"$	The CS policy used is Least Recently Used (LRU)
$c_2(t)$	$m_2(t) * ENTRY\_S \leq m_{13}(t) * 0.8$	The maximum size in memory of the CS is lower than 80% for the system memory
$c_3(t)$	$m_2(t) \leq 10^5$	The size of the CS is smaller than 100000
$c_4(t)$	$m_2(t) \leq m_3(t) * 0.8$	The CS should contain more than 80% of the maximum allowed entries
$c_5(t)$	$std\_dev(m_3([t-4, t])) \leq 5.0$	The standard deviation in number of CS entries is lower than 5.0 for the last 5 iterations
$c_6(t)$	$m_4(t).stdDev \leq 5.0$	The standard deviation in size of the contents stored in the CS is smaller than 5.0
$c_7(t)$	$m_4(t).avg \leq 20.0$	The mean size of contents stored in the CS is greater than 20.0 bytes
$c_8(t)$	$\forall_i\ m_6(t).i < 100$	The entries stored in the PIT are less than 100
$c_9(t)$	$m_7(t).min > 10$	Incoming packets minimum size is greater than 10 bytes
$c_{10}(t)$	$3.0 \leq m_9(t).avg \leq 12.0$	The mean number of components in incoming packets names is within 3.0 and 12.0
$c_{11}(t)$	$m_8(t).min > 10$	Outgoing packets minimum size is greater than 10 bytes
$c_{12}(t)$	$3.0 \leq m_{10}(t).avg \leq 12.0$	The mean number of components in outgoing packets names is within 3.0 and 12.0
$c_{13}(t)$	$\forall_{\text{cert}} \in m_{11}(t) : cert.min \leq t \leq cert.max$	All stored certificates are valid
$c_{14}(t)$	$m_{12}(t) \neq None$	A default certificate is set
$c_{15}(t)$	$m_{14}(t).invalid = 0$	All contents stored in the CS have a valid signature
$r_1(t)$	$c_1(t) \wedge c_2(t) \wedge c_3(t)$	The CS configuration is optimal
$r_2(t)$	$\forall_{t'} \in [t-\delta_t, t]\ c_4(t') \wedge c_5(t') \wedge c_6(t') \wedge c_7(t')$	The CS usage is optimal
$r_3(t)$	$\forall_{t'} \in [t-\delta_t, t]\ c_8(t')$	The PIT usage is optimal
$r_4(t)$	$\forall_{t'} \in [t-\delta_t, t]\ c_9(t') \wedge c_{10}(t')$	The NDN node incoming traffic is optimal
$r_5(t)$	$\forall_{t'} \in [t-\delta_t, t]\ c_{11}(t') \wedge c_{12}(t')$	The NDN node outgoing traffic is optimal
$r_6(t)$	$\forall_{t'} \in [t-\delta_t, t]\ c_{13}(t')$	Certificate status is optimal
$r_7(t)$	$\forall_{t'} \in [t-\delta_t, t]\ c_{14}(t')$	Certificate configuration is optimal
$r_8(t)$	$\forall_{t'} \in [t-\delta_t, t]\ c_{15}(t')$	Cached content signature is optimal

c_{14} verifies whether a content certificate has been set as the default preset on the target NFD node. It can be formally defined as follows:

$$
c_{14}(t) := \begin{cases} \text{True} & m_{12}(t) \neq \emptyset \\ \text{False} & \text{otherwise} \end{cases}
$$

Example 3 (Integrity-Related Composed Rules): Let us consider Example 2 and rules r_6 and r_7 in Table II. r_6 represents the bounds necessary to consider the status of the content certificates stored in the node to be optimal, which

TABLE III
EXAMPLES OF NON-FUNCTIONAL PROPERTIES (SEE DEFINITION 1) CONSIDERING A TIME WINDOW δ_t AND THE CURRENT TIME INSTANT t EXPRESSED IN SECONDS.

Property	***Property Name***	***f(R)***
$p_1(t)$	*Availability*	$\forall_{t'} \in [t-\delta_t, t]\ r_1(t') \wedge r_2(t') \wedge r_3(t') \wedge r_4(t') \wedge r_5(t')$
$p_2(t)$	*Non-repudiation*	$\forall_{t'} \in [t-\delta_t, t]\ r_6(t') \wedge r_7(t')$
$p_3(t)$	*Integrity*	$\forall_{t'} \in [t-\delta_t, t]\ r_6(t') \wedge r_7(t') \wedge r_8(t')$

includes checking that none of them is expired or invalid. r_7 checks the configuration of the node by verifying that a default content certificate has been selected from the ones available. Rule c_{13} returns true if and only if all content certificates stored in the node are valid at time instant t. To measure the stability of this condition we expand the definition using evaluations over time ranges. Thus r_6 can be defined as follows:

$$r_6(t) := \begin{cases} \text{True} & \forall_{t'} \in [t-\delta_t, t]\ c_{13}(t-t) \\ \text{False} & \text{otherwise} \end{cases}$$

Similarly, r_7 and r_8 can be defined by checking respectively c_{14} and c_{15} over the same time range and is successful if and only if all the evaluations are positive.

Example 4 (Properties): Let us consider properties non-repudiation p_2 and integrity p_3 of the data packets produced by the NFD node in Table III. Both properties can be formally described as follows:

$$p_2(t) := \forall_{t'} \in [t-\delta_t, t]\ r_6(t') \wedge r_7(t')$$
$$p_3(t) := \forall_{t'} \in [t-\delta_t, t]\ r_6(t') \wedge r_7(t') \wedge r_8(t')$$

D. Certificate

The certificate is the final outcome of a certification process. It is awarded to the NDN nodes in case of successful evaluation of the set of inference rules related to the given set of properties. It includes the evidence (i.e., measurements and inference rule results) and the corresponding timestamp as follows.

Definition 2: A certificate C is a quadruple $<\mathbf{P},\mathbf{M_t},\mathbf{R_t},t>$ where $\mathbf{P}$ is a finite set of properties p, $\mathbf{R_t}$ is a set of related inference rules, $\mathbf{M_t}$ is a set of measurements supporting $\mathbf{P}$, and t is the time instant of the evaluation.

A certificate can be awarded at time t *iff* $\forall_{p \in \mathbf{P}}$, $p(t).f(R) =$ true. We note that time instant t is important for certificate life cycle management. If a certificate already exists, it is updated every time new evidence is available. In case rule evaluation is negative, the certificate may move to the revoke state.

IV. CERTIFICATION PROCESS

Our certification process efficiently implements the methodology in Section III using a pruning-based approach. When multiple properties are evaluated, corresponding metrics and rules are activated and executed only once, even if requested by multiple rules. This permits to handle complex synchronization between properties requesting different time frames.

We model the set of all possible inference rules and metrics as a DAG where:

- each node represents a task;
- if an arch is exiting from a node A and entering in a node B, we say that the execution of A requires the results of the execution of B;
- metrics cannot have dependencies, that is, their nodes can only have incoming arcs;
- rules can depend both on rules or measurements retrieved by metrics.

Property certification is defined over the DAG and consists of the following steps.

DAG Pruning. It prunes those nodes in the DAG that are not required for the evaluation of the given properties. We then define a function that selects all graph nodes that are relevant for the evaluation. With $r(n)$ being the set of nodes required by node n, the selection function s can be defined recursively as follows:

$$s(n) = \{n\} \cup \bigcup_{d \in r(n)} s(d)$$

The graph of all the possible computable tasks is reduced to the union of the sets produced by applying the selection function to each property. With $\mathbf{T}$ being the set of tasks in the graph and $\mathbf{P} \subseteq \mathbf{T}$ the set of properties we want to validate, we can compute $\hat{\mathbf{T}}$, the minimal set of tasks to evaluate as follows:

$$\hat{\mathbf{T}} = \bigcup_{n \in \mathbf{P}} s(n)$$

This solution uses the naive assumption that all the required tasks output are necessary to complete the task evaluation. A more efficient implementation can differentiate the selection based on the content of the task received as input.

Tasks execution. This step executes each task of the pruned graph, such that all of its requirements have been evaluated beforehand. Task execution is described using the pseudocode in Algorithm 1. Algorithm 1 shows how to iterate the pruned set of tasks to check if all the dependencies have been resolved. If yes, the evaluation starts; otherwise, the task is set aside until the next iteration loop. The lack of cycles in the DAG allows us to assume that each task will eventually have all its dependencies resolved. We note that when the evaluation of a rule or a metric fails (e.g., for a network connection error or missing data), an error is triggered. Depending on the rule definition, such error may not necessarily cause an abort of the whole evaluation.

Property evaluation. It collects the outcomes of each task execution to evaluate the properties. For each property, the CA checks whether the corresponding rule evaluation is successful. If all property evaluations return a positive result, the CA awards the corresponding certificate.

Example 5: Let us consider the evaluation of properties p_2 and p_3, in our previous Example 4, which refers to properties non-repudiation and integrity, respectively. Figure 3(a) summarizes the dependency relationships between the interested tasks.

Algorithm 1 Tasks execution

Input: tasks

```
1: to_execute = tasks
2: while to_execute ≠ ∅ do
3:    for task ∈ to_execute do
4:       can_be_executed = true
5:       for dep ∈ dependencies(task) do
6:          if dep ∈ to_execute then
7:             can_be_executed = false
8:             break
9:          end if
10:      end for
11:      if can_be_executed then
12:         execute(task, dependencies(task))
13:         to_execute = to_execute \ {node}
14:      end if
15:   end for
16: end while
```

Algorithm 2 Property evaluation

Input: tasks, properties

```
1: for task in tasks do
2:    if task in properties and not succeeded(task) then
3:       return task error
4:    end if
5: end for
6: release_certification()
```

Our certification process first prunes the certification graph by selecting only the tasks that are necessary for the evaluation of the given properties. In this case, the evaluation of the set $\hat{\mathbf{T}}$ starting from $\mathbf{T} = \{p_2, p_3\}$ is the following:

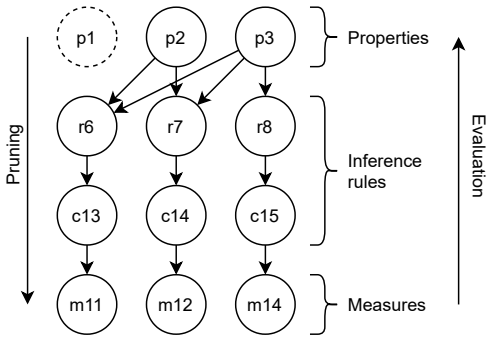
$$\begin{aligned}\hat{\mathbf{T}} &= s(p_2) \cup s(p_3) \\ &= \{p_2, p_3\} \cup s(r_6) \cup s(r_7) \cup s(r_8) \\ &= \{p_2, p_3, r_6, r_7, r_8, c_{13}, c_{14}\} \cup \\ &\quad s(c_{13}) \cup s(c_{14}) \cup s(c_{15}) \\ &= \{p_2, p_3, r_6, r_7, r_8, c_{13}, c_{14}, c_{15}\} \cup \\ &\quad s(m_{11}) \cup s(m_{12}) \cup s(m_{14}) \\ &= \{p_2, p_3, r_6, r_7, r_8, c_{13}, c_{14}, c_{15}, m_{11}, m_{12}, m_{14}\}\end{aligned}$$

We recursively apply function s to the initial set $\mathbf{T}$. At each application step, we show how the group contains the task as input and the result of the function s applied to each of its dependencies. The function stops when we reach a task that has no dependencies, in this case the metrics m_{11}, m_{12} and m_{14}. Figure 3(b) shows the DAG after the pruning step when evaluating the properties p_2 and p_3.

It then executes the first non-blocked tasks, that is, the metrics m_{11}, m_{12}, and m_{14}. The collected measurements are stored and made available to the next tasks, that is, rules c_{13},c_{14} and c_{15}, which are ready to be executed since all their requirements completed. Rules r_6, r_7, and r_8 describe a requirement asking consecutive validity of their dependencies

Id	Dependencies
c_{13}	m_{11}
c_{14}	m_{12}
c_{15}	m_{14}
r_6	c_{13} in last δ_t s
r_7	c_{14} in last δ_t s
r_8	c_{15} in last δ_t s
p_2	$r_6 \wedge r_7$ in last δ_t s
p_3	$r_6 \wedge r_7 \wedge r_8$ in last δ_t s

(a)



(b)

Fig. 3. Table of dependencies (a), Pruned DAG (b)

for at least 2 minutes, meaning that their output will be true *iff* all the evaluations of their dependencies in the last 2 minutes have given a positive result. Depending on the strictness of the bounds, we can require that all the dependency evaluations within 2 minutes have produced a positive results or, more rigorously, that at least two minutes worth of measurements have been attempted and have generated a positive output. The same distinction can be done using the number of iterations, that is, a certain task had only positive results in the last 5 iterations. Similarly, both properties p_2 and p_3 require rules r_6 and r_7 to be positively evaluated for 2 minutes, with p_3 also requiring r_8. Once p_2 and p_3 have been executed, all the tasks in $\hat{\mathbf{T}}$ have been completed and the certification process can start the collection of the property evaluation results. If and only if all the properties have returned a positive output, the CA awards a certificate as output. In the event of an input/output error during the measures or a missing dependency in a rule evaluation, the process interrupts immediately and returns a report of the execution state.

V. Experimental Evaluation

We experimentally evaluated the soundness and performance of our certification approach showing how the measurements collected by our metrics can be used to *i)* identify a change in the system state and *ii)* evaluate inference rules that permit to issue/confirm a certificate in case of positive evaluation or revoke it, otherwise. We first present our experimental setup; we then provide a complete certification walkthrough to show the utility and usability of our approach; we conclude

with a performance evaluation. Interested readers can access all results at *https://bit.ly/3xPDmiy*.

A. Experimental setup

We run our experiments in a controlled and repeatable environment based on Mini-NDN[2], an extension of the network virtualization system Mininet[3] specific for NDN. Mini-NDN permits to deploy a set of virtualized NDN nodes on a single host and to run our software. In particular, to generate realistic traffic in the network, we used a producer-consumer paradigm installing the following services on the nodes:

- *Producer:* a NDN service that listens for interest requests on a specific prefix and returns a data packet with a random string of fixed length. Having a unique prefix for each node, these services can be queried by any NDN clients in the network. Multiple producers can be deployed on the same node on different prefixes.
- *Consumer:* a NDN service that repeatedly requests content from a specific prefix domain. The rate of the requests and the used domains can be customized to better simulate traffic using a list of known content paths, sampled with a linear or Zipf-like probability distribution.
- *CA:* an agent running in the target node and supporting our certification methodology. It checks for updates on the certificate status every second evaluating relevant inference rules.

The network topology is a three-node chain, where each node uses the base NFD configuration, enabling the caching of unsolicited data and limiting the CS to 300 entries. The two outer nodes generate network traffic using a two-way, producer-consumer connection: each node registers a unique prefix for its producer service, responding to any requests with a random string of length 20. The same nodes also run a consumer service, requesting contents from the other node prefix. The consumer uses a domain of content names obtained converting the NASA website HTTP Requests data set to the NDN protocol, released along the application source code, a Zipf-like distribution with $s = 1.2$, and a request rate of $5/s$.

The default implementation of the NFD process collects various metrics on its internal status through specific tools (*nfdc* and *ndnsec*) producing either a status report in XML format or a parseable output. These tools are shipped within the NFD distribution and do not interfere with the protocol. To capture our certification measurements, we add new metrics to the *nfdc* status report and use *ndnsec*. These tools are already deployed within the NFD distribution and we extend them in a compatibility-preserving manner.

The experiments were executed on a Linux kernel 5.11 zen machine equipped with Intel i7-8750h, 32GB of RAM and NVME SSD. They targeted properties p_1 (i.e., availability), p_2 (i.e., non-repudiation), and p_3 (i.e., integrity), using the complete rule set described in Section IV and a certification time window δ_t of 120s.

[2]https://github.com/named-data/mini-ndn

[3]https://github.com/mininet/mininet

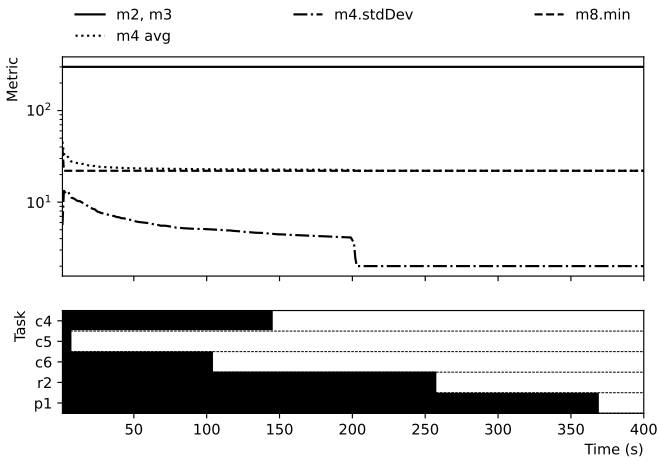


Fig. 4. Certification of property p_1 with the more relevant measurements and inference rules. White and black horizontal bars mean positive or negative evaluation respectively.

B. Certificate Awarding Process

We simulated a normal and almost constant NDN traffic to show the certificate awarding process in action. After the first evaluation time frame, properties p_2 and p_3 were immediately certified due to the NDN default settings discussed in Section V-A. In the case of property p_1, a longer period of time was needed to aggregate metrics to evaluate the relevant inference rules. Figure 4 shows with a two-section chart the evolution of measurements (i.e., m_2, m_3 and m_4 at the top with a line plot) and rules (r_2, c_6, c_5 and c_4 at the bottom with an horizontal bar chart) relevant for property p_1 (plotted as well in the bottom bar chart). Figure 4 shows, for each time instant, the value of the selected measurements (top), the outcome of each rule/property (bottom), that is, positive outcome in white or negative outcome in black. We note that, for clarity, we do not report those measurements and rules that are always providing a positive outcome. We also note that the same plot style is used in the remaining of these experiments.

The top line chart of Figure 4 shows the output of metrics using a logarithmic scale. As expected we can see how the number of packets stored in the CS gradually grew until the maximum level, reported by m_2, was reached at t=200s. The metric m_4, measuring statistics about the size in memory of the cached contents, shows how the standard deviation was slowly descending, until t=203s, where it dropped to approximately 2. This is due to the fact that almost the totality of the data packets came from the outer node producers. The bottom horizontal bar chart of Figure 4 shows how the most significant rules varied during the experiment. Rule c_4, which requires the CS to contain at least 80% of its maximum number of entries, returned a positive output starting from t=147s; c_5, which requires the CS stored contents standard deviation to be smaller than 5 in the last 5 iterations, was rapidly reached at t=9s; c_6, which requires the stored contents size standard deviation to be smaller than 5, was reached at t=106s. Rule r_2,

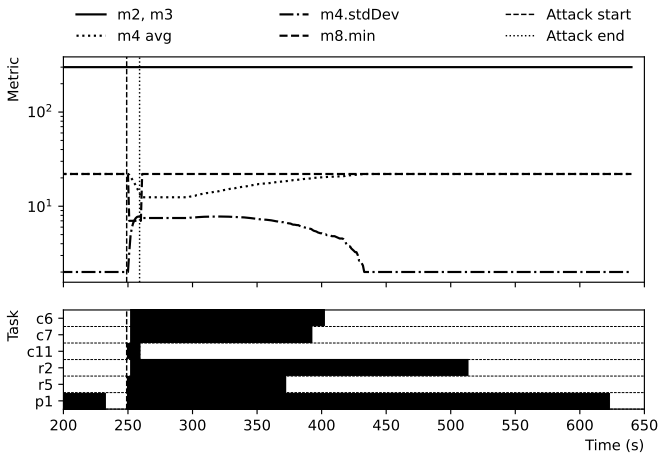


Fig. 5. CAation for property p_1 in case of cache pollution attack with the more relevant measurements and inference rules.

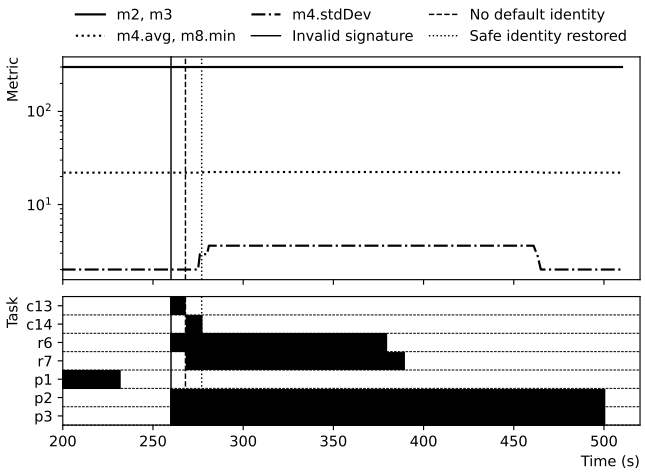


Fig. 6. Certification revocation for property p_2 and p_3 in case of misconfiguration with the more relevant measurements and inference rules.

which requires all its dependencies to return positive outputs in the last two minutes, was negative until t=259s. At time t=370s, p_1 became valid since all dependencies including r_2 were valid. CA finally awarded the certificate to the NDN node.

C. Certificate Revocation

Let us consider a cache pollution attack where a producer cooperates with the attacker by replying to any received requests. The attack was simulated starting an additional pair producer-consumer in the adjacent nodes using a unique prefix. The producer returned random strings of five characters, while the consumer used a linear probability distribution over the NASA data set and a request rate of $20/s$.

The attack was simulated for approximately 10 seconds with the goal of testing certificate revocation for property p_1. The entire experiment continued until the certificate was restored for the target node. Figure 5 shows the experiment results using our two-section chart. The certificate for p_1 was awarded from t=235s. The attack traffic started at t=250s. The measurements immediately showed the effect of the attack: the average of m_4 dropped down to 12.45 at t=261s, while the standard deviation peaked at 7.77 at t=321s. m_8, measuring the outgoing data packets size statistics, dropped its minimum to 7 as soon as the attack started. This strong alteration of the NDN was immediately identified by the certification within few seconds after the beginning of the attack, and the corresponding certificate was revoked. The attack ended after 10 seconds and the system slowly recovered, finally being re-assigned with a certificate for p_1 at t=624s.

Let us now consider a misconfiguration attack violating properties p_2 and p_3. For the misconfiguration procedure, we created a new identity, signed its certificate with a past validity time frame, and set it as default. The system default identity then used an expired certificate. After 10 seconds we removed the new certificate, leaving the node without a default certificate. Finally, after 10 more seconds, we reset the default identity to the initial valid one, and waited for the system to return to a certified state.

Figure 6 shows the experimental results using our two-section chart. The certificate for p_1, p_2, and p_3 was awarded at t=234s.

The misconfiguration attack effects were immediately visible at t=260s, where the rule c_{13} was violated due to the presence of an invalid identity on the NDN node. This also reflected in the violation of rule r_6, and the corresponding properties p_2 and p_3 resulting in certificate revocation. After 10 seconds, the invalid default identity was removed, resulting in r_6 producing a positive outcome and r_7 a negative one. After 10 more seconds, the original valid identity was set as default, reverting the r_7 outcome to a positive one. After 2 more minutes (δ_t time window) the certificate for properties p_2 and p_3 was restored.

D. Performance evaluation

To measure the performance of our certification approach we adopted the same simulation settings used in Section V-B based on normal traffic. We measured the time needed to obtain the measurements, verify the inference rules and properties, and issue the corresponding certificates. Figure 7 shows the execution time of the certification process averaged over 5 simulations varying the properties involved in the certification. We note that the certification process related to property p_1 was more computational intensive than the process needed to certify p_2 and p_3 together. This effect is due to the number of rules involved in p_1 compared to the rules involved in p_2 and p_3. We also note that the computation time linearly increased across time. This effect is due to the incremental number of evidence accumulated across time that needs to be considered for the certification. In general, even with frequent certification attempts (one per second), in the worst case with all the three properties and after 500 certificate iterations, the

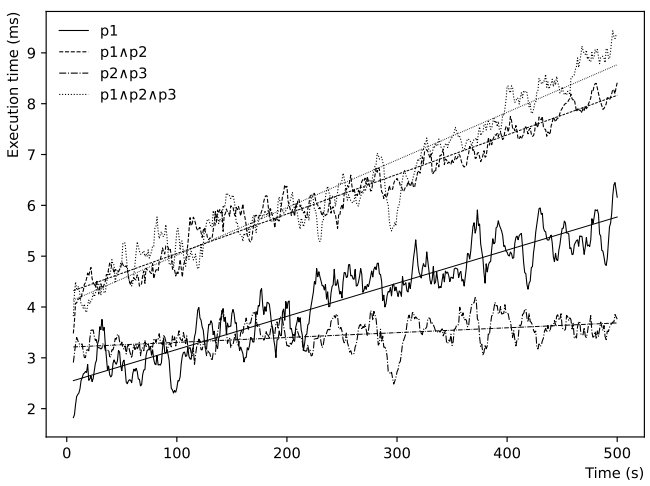


Fig. 7. Certification process performance considering different properties and the scenario in Section V-B

computation effort was lower than 10ms. Furthermore, the maximum memory usage measured was lower than 150MB. In specific constrained conditions, by applying a pruning step after each cycle, we could cap the growth of the log size, removing old and unused information, thus reducing both the execution time and the memory usage to values almost constant during the execution.

VI. Related Work

Existing approaches for NDN evaluation have focused primarily on identification and mitigation of security attacks [10]–[14]. Yao et al. [10] described a cache pollution prevention system based on gray prediction models, differentiating between false-locality and locality-disruption attacks against contents popularity. Salah et al. [11] proposed a modification of the NDN protocol, permitting to exactly measure the popularity distribution of any contents in a trusted network. Karami and Guerrero-Zapata [12] used a similar approach, adapting an ANFIS neural network to the analysis of cached contents information.

More recently, research has focused on monitoring of content-based networks traffic with the scope of evaluating both networking nodes and protocols [14], [19], [20]. Zhou et al. [19] described a network-behavior monitoring solution focused on the identification of congestion. Bialas et al. [20] summarized the modern network monitoring techniques and its application for anomaly detection. Nguyen et al. [14] focused on the monitoring of NDN traffic to identify common attacks against the network using distributed probes.

In this paper, we proposed a certification methodology with the scope of improving trustworthiness in NDN. Our approach can integrate with existing NDN monitoring techniques to effectively prove non-functional properties. Certification methodologies have been successfully applied to improve trustworthiness and transparency in many application context [18], [21]–[25]. Our certification methodology, similarly to the service-based ones [21], [24], focused primarily on the software components and on the verification of security mechanism behavior. To the best of our knowledge, our paper is the first attempt to develop a certification methodology for NDN.

VII. Conclusions

Trustworthiness and transparency are fundamental to unleash the full potential and increase the adoption of content-centric networks. Current literature has mainly proposed protocol extensions or monitoring techniques to prevent or mitigate specific security attacks. In this paper, we proposed an evidence-based non-functional property certification methodology for NDN networks that can complement existing monitoring techniques increasing the overall network trustworthiness. Our certification process implements a rule-based behavioral analysis grounded on network measurements captured in operation. The results of our experimental evaluation show the feasibility of our methodology. We believe this paper can pave the way to novel QoS-oriented content-centric networks centered on the non-functional properties of the involved nodes.

Acknowledgment

Research supported, in parts, by EC H2020 Project CONCORDIA GA 830927, Università degli Studi di Milano under the program "Piano sostegno alla ricerca". Filippo Berto acknowledges support from TIM S.p.A. through the PhD scholarship.

References

[1] 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.

[2] A. Afanasyev, J. Burke, T. Refaei, L. Wang, B. Zhang, and L. Zhang, "A brief introduction to named data networking," in *Proc. of MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. Los Angeles, CA: IEEE, 2018, pp. 1–6.

[3] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, and A. Ksentini, "Named data networking in vehicular ad hoc networks: State-of-the-art and challenges," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 320–351, 2020.

[4] Z. Li, Y. Liu, Y. Chen, Y. Xu, and K. Liu, "Performance analysis of a novel 5g architecture via content-centric networking," *Physical Communication*, vol. 25, pp. 328–331, 2017.

[5] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, and L. Zhang, "Named data networking of things (invited paper)," in *Proc. of 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, Berlin, Germany, 2016, pp. 117–128.

[6] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks," *IEEE Network*, vol. 28, no. 6, pp. 91–96, 2014.

[7] C. Tsilopoulos and G. Xylomenos, "Supporting diverse traffic types in information centric networks," in *Proc. of ACM SIGCOMM workshop on Information-centric networking*, ser. ICN '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 13–18.

[8] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking," in *Proc. of 2013 IFIP Networking Conference*, Brooklyn, NY, USA, 2013, pp. 1–9.

[9] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS and DDoS in named data networking," in *Proc. of 2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, Nassau, Bahamas, 2013, pp. 1–7.

[10] L. Yao, Y. Zeng, X. Wang, A. Chen, and G. Wu, "Detection and defense of cache pollution based on popularity prediction in named data networking," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.

[11] H. Salah, M. Alfatafta, S. SayedAhmed, and T. Strufe, "CoMon++: Preventing cache pollution in NDN efficiently and effectively," in *Proc. of 2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. Singapore: IEEE, 2017, pp. 43–51.

[12] Amin Karami and Manel Guerrero-Zapata, "An ANFIS-based cache replacement method for mitigating cache pollution attacks in named data networking," *Computer Networks*, vol. 80, pp. 51–65, 2015.

[13] 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.

[14] T. Nguyen, H. Mai, G. Doyen, R. Cogranne, W. Mallouli, E. M. d. Oca, and O. Festor, "A security monitoring plane for named data networking deployment," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 88–94, 2018.

[15] T. Nguyen, X. Marchal, G. Doyen, T. Cholez, and R. Cogranne, "Content poisoning in named data networking: Comprehensive characterization of real deployment," in *Proc. of 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, Lisbon, Portugal, 2017, pp. 72–80.

[16] T. Liang, J. Pan, M. A. Rahman, J. Shi, D. Pesavento, A. Afanasyev, and B. Zhang, "Enabling named data networking forwarder to work out-of-the-box at edge networks," in *Proc. of 2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Dublin, Ireland, 2020, pp. 1–6.

[17] M. Hussaini, S. A. Nor, H. Bello-Salau, H. J. Hadi, A. A. Gumel, and K. A. Jahun, "Mobility support challenges for the integration of 5g and IoT in named data networking," in *Proc. of 2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf)*, Zaria, Nigeria, 2019, pp. 1–7.

[18] M. Anisetti, C. A. Ardagna, F. Gaudenzi, and E. Damiani, "A certification framework for cloud-based services," in *Proc. of 31st Annual ACM Symposium on Applied Computing - SAC '16*. Pisa, Italy: ACM Press, 2016, pp. 440–447.

[19] Y. Zhou, J. Bi, T. Yang, K. Gao, J. Cao, D. Zhang, Y. Wang, and C. Zhang, "HyperSight: Towards scalable, high-coverage, and dynamic network monitoring queries," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1147–1160, 2020.

[20] A. Bialas, M. Michalak, and B. Flisiuk, "Anomaly detection in network traffic security assurance," in *Proc. of Engineering in Dependability of Computer Systems and Networks*, ser. Advances in Intelligent Systems and Computing, W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, and J. Kacprzyk, Eds. Cham: Springer International Publishing, 2020, pp. 46–56.

[21] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Gaudenzi, "A semi-automatic and trustworthy scheme for continuous cloud service certification," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 30–43, 2020.

[22] S. Lins, S. Schneider, J. Szefer, S. Ibraheem, and A. Sunyaev, "Designing monitoring systems for continuous certification of cloud services: Deriving meta-requirements and design guidelines," *Communications of the Association for Information Systems*, vol. 44, no. 1, 2019.

[23] S. Lins, S. Schneider, and A. Sunyaev, "Trust is good, control is better: Creating secure clouds by continuous auditing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 890–903, 2018.

[24] M. Anisetti, C. A. Ardagna, E. Damiani, N. El Ioini, and F. Gaudenzi, "Modeling time, probability, and configuration constraints for continuous cloud service certification," *Computers & Security*, vol. 72, pp. 234–254, 2018.

[25] I. Kunz and P. Stephanow, "A process model to support continuous certification of cloud services," in *Proc. of 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, Taipei, Taiwan, 2017, pp. 986–993.