Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

Enhancing Privacy in Internet of Things using Software Defined Network

Shahrbanoo Zangaraki , ***Seyed Hossein Erfani*** * , ***Amir Sahafi***
Department of Computer Engineering, South Tehran Branch, Islamic Azad University, Tehran, Iran.

Article Info

Article History:
Received 09 June 2025
Reviewed 11 August 2025
Revised 07 September 2025
Accepted 18 September 2025

Keywords:
IOT
SDN
Privacy Preserving

*Corresponding Author's Email Address: *h_erfani@azad.ac.ir*

Abstract

Background and Objectives: The Internet of Things (IoT) serves as a fundamental communication model, enabling objects to deliver data and services to users. With the rapid expansion of IoT, ensuring privacy and preventing the disclosure of sensitive data during message exchanges between objects has become increasingly challenging. This paper presents an attribute-based framework designed to enhance privacy protection in IoT environments by leveraging software-defined networking (SDN) technology.
Methods: By leveraging the SDN and the Attribute-Based Privacy Preserving (ABPP) model, our proposed framework employs an advanced algorithm to enhance privacy for client requests accessing IoT services. It focuses on protecting sensitive information during message transmission by implementing techniques for anonymity, unlinkability, and untraceability, tailored to the sensitivity level of each message. To further enhance message privacy within the IoT network, our framework incorporates IP aliasing, dynamic channel switching, and payload encryption.
Results: Our proposed framework significantly enhances privacy protection in IoT networks by dynamically applying anonymity and concealment techniques tailored to the sensitivity of CoAP messages. Simulation results using CloudSimSDN confirm the framework's effectiveness in safeguarding sensitive information while maintaining optimal communication performance. Using three privacy-preserving methods leads to an average CPU utilization that is 0.14 units higher than when only one method is applied. We provide a security evaluation that includes formal verification techniques and informal analysis, and show that the proposed framework is secure against anonymity and Man in The Middle (MITM) attacks, replay attacks, Sybil, and IP spoofing.
Conclusion: In this paper, we present a four-layer SDN-based framework designed to enhance privacy in IoT networks through the use of the Attribute-Based Privacy Preserving (ABPP) model. The framework employs IP aliasing, dynamic routing, and content encryption techniques tailored to the sensitivity of CoAP messages to ensure data protection. Our implementation and experiments conducted with CloudSimSDN validate the framework's effectiveness in safeguarding sensitive information.

How to cite this paper:
S. Zangaraki, S. H. Erfani, A. Sahafi, "Enhancing privacy in internet of things using software defined network," J. Electr. Comput. Eng. Innovations, 14(1): 163-180, 2026.
DOI: 10.22061/jecei.2025.11621.821
URL: https://jecei.sru.ac.ir/article_2404.html

Introduction

Today, numerous aspects of human life are influenced by the Internet of Things (IoT). The IoT is utilized across various domains, including agriculture, patient monitoring, home automation, welfare, and smart cities, among others. Experts predict that by 2030, the number of connected devices within IoT networks will exceed 29 billion [1], [2]. Privacy entails that information about individuals must be safeguarded and not disclosed under any circumstances without explicit consent. It also ensures that users' identities cannot be discerned or tracked based on their behavior and actions within the system. The vast amount of data generated by billions of Internet of Things (IoT) devices poses a significant threat to user privacy, heightening the risk of breaches and privacy violations [2]-[5].

Ensuring privacy in modern systems has become increasingly important. The primary objectives of privacy techniques are to achieve anonymity, unlinkability, and untraceability. Anonymity ensures that a third party cannot identify an individual's identity among other identities within the system. Unlinkability refers to the inability to associate a person's identity with the information they produce. Untraceability means that it is difficult to track actions and information generated from an entity's behavior within the system [2], [4].

A new technology, known as SDN, has been introduced in the networking industry. Its primary purpose is to decouple the control logic from the network equipment, such as transport devices. This separation allows for the implementation of control logic on physical devices based on the specific requirements of the application. SDN comprises three layers: data, control, and application. The application layer connects to the control layer via the northbound interface, while the control layer connects to the data layer via the southbound interface [6]-[8].

The use of SDN in computing systems to manage the Internet of Things (IoT) offers several advantages, including flexibility, scalability, redundancy, and reduced hardware requirements. SDN enables users to achieve greater flexibility in their operations and architecture, which is particularly important for IoT system architecture. The use of distributed infrastructure and limited resources in IoT applications further underscores the importance of this flexibility. As SDN is increasingly adopted in IoT applications, privacy challenges must be addressed. Proper software design and the implementation of various applications are necessary to mitigate these challenges. Although significant progress has been made in privacy protection for IoT, ensuring privacy in machine-to-machine (M2M) IoT networks remains a challenge. [4], [2], [9], [7], [10], [11].

This paper proposes ABPP-SDN, an IoT attribute-based privacy protection framework that integrates encryption, anonymity, and dynamic channels using Software-Defined Networking (SDN) technology. The objective is to enhance the privacy of IoT networks by leveraging SDN as the network infrastructure. To facilitate message transfer between different devices, ABPP employs the Constrained Application Protocol (CoAP), a lightweight and widely used web-based protocol in IoT networks. CoAP [12], [13] supports resource discovery, block transfer, and asynchronous message exchange between devices, but it lacks advanced privacy enforcement mechanisms-such as dynamic routing and attribute-based encryption-that are essential for fine-grained protection in sensitive applications. To define different levels of privacy for IoT services, ABPP-SDN adopts an attribute-based privacy model. This framework aims to address the limitations identified in existing research and underscores the importance of IoT privacy. ABPP-SDN introduces a new component for the SDN controller to apply privacy protection levels during message transmission, thereby achieving privacy protection goals. The proposed solution involves using a new alias to anonymize the source address during transmission. Additionally, encryption, dynamic channels, padding, and compression techniques are employed to conceal sensitive information, ensuring untraceability and unlinkability, However, while these layered techniques enhance privacy guarantees, they also introduce non-negligible computational and communication overhead. In summary, the main contributions of this work include:

- We introduce our proposed framework, a four-layer SDN-based architecture designed to enhance privacy protection and safeguard sensitive information during message transmission among devices in IoT networks.
- We define ABPP, an attribute-based privacy-preserving model, ABPP-SDN, to ensure the privacy of sensitive data during message transmission within the IoT network. This model considers the varying degrees of sensitivity in CoAP messages. We quantify the sensitivity level of each CoAP message and select appropriate anonymization and concealment techniques based on its respective sensitivity before transmission.
- We simulate our proposed framework using CloudSimSDN and conduct several experiments to evaluate its performance.

Background

This section provides a concise overview of the Software-Defined Networking (SDN) paradigm and the Constrained Application Protocol (CoAP).

A. SDN Technology

SDN is an innovative technology that enhances network efficiency by decoupling the control plane from the data plane. This separation facilitates improved network management and addresses challenges such as security, scalability, heterogeneity, and limited capacity within the Internet of Things (IoT) [7], [4], [14].

As illustrated in Fig. 1, SDN comprises three layers: the application layer, the control layer, and the data layer [15], [7]. Each layer communicates with its adjacent layer through a set of interfaces. The application layer contains SDN applications that define network behavior via the northbound interface. The most prominent northbound interface is the REST API, which allows remote applications to send commands to or retrieve information from the controller using the HTTP protocol. The control layer consists of one or more logically centralized SDN controllers responsible for managing the control plane and creating a network view. The controller's role is to translate application requests into data plane instructions via the southbound interface and provide an updated network state to the applications. The data layer comprises devices such as switches and routers, which are responsible for forwarding packets. Communication between the data layer and the control layer is facilitated through the southbound interface, typically implemented via the OpenFlow protocol. Consequently, SDN is often referred to as SDN/OpenFlow [16], [8].

B. CoAP Protocol

The CoAP protocol was developed by the IETF as an application layer protocol for IoT applications based on the REST architecture [13], [12]. Because most IoT devices have limited resources, including memory and processing capability, HTTP is not well suited due to its complexity. As a result, the CoAP protocol was developed by adapting and simplifying specific features of HTTP. The key characteristics of the CoAP protocol include the following:

This protocol is designed for web usage and implements a request/response model for limited resources.

- Asynchronous communication is supported for message passing.
- This protocol has lower overhead and is less complex than HTTP.
- This feature provides assistance for various types of content and Uniform Resource Identifiers (URIs).
- This feature facilitates the process of finding and utilizing services and resources, and also enables communication through multicast.
- This protocol offers a straightforward proxy feature and is capable of interacting with the HTTP protocol.
- The CoAP token field enables correlation between request and response.

When evaluating lightweight protocols for IoT communication, CoAP stands out against alternatives such as Message Queuing Telemetry Transport (MQTT) [17] and Google Remote Procedure Calls (gRPC) [18] due to several architectural and functional distinctions [19] :

IoT uses **CoAP** in resource-constrained devices with RESTful interactions that use web services. Features to support multicast communication and integration with proxies are also offered. The primary advantages of CoAP include a lightweight design coupled with a built-in DTLS, serving privacy and security. The disadvantages come from missing large sets of mechanisms: for example dynamic routing and attribute-based encryption can be a major constraint for privacy-sensitive applications.

MQTT is a publish-subscribe pattern well-suited for applications where reliability is a major concern as it provides QoS (Quality of Service) levels. However, it is broker-based and hence may need an external QoS for multicast communications and RESTful interactions. Provision of privacy in MQTT requires external methods and manual configuration of TLS, often quite complex in itself.

gRPC has the power and efficiency required from any high-performance protocol. It is based on HTTP/2 and uses Protocol Buffers promising high data-transfer efficiency. However, all these merits come with the price of high overhead signatures imposed on the underlying system making it inappropriate for use in resource-constrained IoT systems. Furthermore, it misses out on providing any built-in privacy at the protocol level, which can be significantly detrimental for sensitive applications.

Related Work

Ensuring privacy protection is a critical requirement in IoT environments [2], [20]-[23]. IoT networks involve the exchange of sensitive data and the potential for unauthorized access, making privacy preservation imperative. While conventional privacy protection methods face challenges when applied to resource-constrained IoT devices, SDN technology provides promising solutions to address privacy concerns. By leveraging SDN's centralized control and programmability, privacy protection methods can be implemented at the network level. In this section, we review the existing literature on privacy protection methods for IoT networks based on SDN technology.

A context-aware privacy-preserving method is presented in [24] for IoT-based smart cities using SDN technology. Their method involves splitting data packets into different packets and managing them based on their

contexts and privacy levels. However, the method has some limitations. First, it requires a centralized controller to manage data packets, which may introduce a single point of failure and create a bottleneck. Second, it does not account for the dynamic nature of IoT devices' mobility and connectivity, which could affect the context awareness and privacy levels of the data packets.

In [25], a privacy protection method is presented to mitigate the risk of data leakage in IoT-SDN networks by periodically altering the switch-controller mapping. However, it is important to note that this method overlooks the heterogeneity, topology, and traffic aspects of IoT networks.

In [26], a presented edge computing-enhanced IoT framework is introduced to enhance privacy preservation in smart cities. This framework identifies two key challenges related to IoT data management: (1) the heterogeneity of IoT devices, and (2) the preservation of sensitive data privacy. To address the heterogeneity challenge, the framework adopts an ontology-based data model that captures crucial information about IoT devices and their corresponding privacy levels. Additionally, to tackle the privacy preservation challenge, the framework presentes a method that periodically adjusts the privacy-preserving behaviours of IoT devices based on insights derived from the ontology.

The method mentioned in [27], IoT-SDNPP, uses SDN technology for the privacy preservation aspect in smart cities. With IoT-SDNPP, SDN is used to separate the control plane from the data plane to make the management and flexibility more applicable to the network. With these privacy-preserving rules taking into consideration every individual IoT device's specific characteristics and environment, it aims to provide totally personalized protection in the privacy arena for the IoT ecosystem.

Reference [28] introduces a novel privacy-preserving approach for IoT networks based on differential privacy. It demonstrates how differential privacy can be utilized to safeguard data generated by IoT devices within an IoT ecosystem. The proposed method assesses the sensitivity of the data and injects noise into the relevant data dimensions accordingly, ensuring privacy protection while maintaining data utility. As a result, the approach is regarded as a robust and flexible solution for enhancing privacy in IoT networks.

In [29], an advanced privacy and functional authentication scheme is introduced, specifically designed for fog nodes in smart healthcare. This scheme leverages the capabilities of SDN technology to implement an efficient authentication mechanism within the SDN gateway. The primary objective is to validate the credibility of fog nodes while minimizing the computing overhead of IoT devices. By utilizing privacy and functional attributes, the scheme ensures the secure authentication of both fog nodes and IoT devices. This innovative approach offers enhanced privacy protection and reliable authentication mechanisms for fog computing in the context of smart healthcare applications.

In [30], a dynamic privacy-preserving method based on SDN is presented for smart cities, incorporating a trust technique. This method leverages SDN technology to deploy a mechanism within the SDN controller. The mechanism operates based on the mutual trust among nodes and dynamically selects different routes from IoT devices to the cloud environment, depending on the confidence level. Additionally, the SDN controller reroutes packets when it detects a device that lacks trust in its neighbouring device. This innovative approach ensures privacy protection while optimizing network routing and enhancing trustworthiness within the smart city infrastructure.

A layered architecture, termed ESDNS-DLHFS, is presented in [31], incorporating privacy enhancement and intrusion detection mechanisms within SDN-based IoT networks. The framework integrates min-max normalization with a hybrid Crow Search–Arithmetic Optimization approach for feature selection, and employs Deep BiLSTM for attack detection, alongside an enhanced Artificial Orca's algorithm for hyperparameter optimization. It is designed to achieve a balance between privacy protection and computational efficiency by leveraging deep learning and bio-inspired algorithms to secure consumer IoT platforms. This study exemplifies the integration of machine intelligence into SDN-driven security architectures and contributes to advancing methodologies for privacy preservation in resource-constrained environments.

[32] introduces a smart, new approach to protecting privacy in smart cities built on IoT, using the SDN technology. The method assesses trust levels among neighbouring devices to safeguard the privacy of IoT devices. The authors compared their approach with existing methods and found that, although it uses a bit more resources, it's more effective at preventing accidental data leaks—especially in situations where adversaries might have some background knowledge.

As depicted in Table 1, the proposed framework significantly advances privacy protection in IoT networks by introducing a four-layer SDN-based architecture that enhances the protection of sensitive information during device communication. Unlike previous approaches, our framework incorporates an attribute-based privacy preservation (ABPP) model, ABPP-SDN, which dynamically adjusts protection measures based on the

sensitivity of CoAP messages. By quantifying the sensitivity degree of each message and applying appropriate anonymization and concealment techniques, our approach ensures a higher level of privacy protection, addresses the limitations of existing studies, and provides a more dynamic privacy protection system for IoT networks.

Table 1 : Comparison of exiting work

Related Work	Properties of Privacy Protection Method			Architecture Type
	Support Data Fragmentation	Context-Aware	Dynamic	SDN-Based
ABPP	Yes	Yes	High	Yes
[24]	Yes	Yes	Medium	Yes
[25]	No	Yes	Medium	Yes
[26]	No	Yes	Medium	NO
[27]	Yes	Yes	Medium	Yes
[28]	No	NO	Medium	Yes
[29]	No	Yes	Medium	Yes
[30]	Yes	Yes	Medium	Yes
[31]	No	Yes	Medium	Yes
[32]	Yes	Yes	Medium	Yes

ABPP-SDN Framework

In this section, we will provide an overview of ABPP-SDN, a system designed for attribute-based privacy protection. We will first explain the system architecture and then describe the ABPP model that is used to determine the level of privacy protection. In addition, we will discuss methods used to ensure different levels of privacy for CoAP response messages during message transmission. We will also discuss methods to avoid network analysis.

A. Architecture

Generally, the system architecture of our proposed framework as depicted in Fig. 1, which incorporates SDN-IOT [33], [34], consists of four layers, as illustrated in Fig. 1. This architecture enables various CoAP servers to provide diverse CoAP services within the IoT network, which can be accessed by CoAP clients through IoT applications. The layers of our proposed framework are as follows:

Application Layer: This layer hosts the IoT applications that provide different CoAP services to the CoAP clients. A CoAP client can register and request access to the CoAP services via the IoT application through this layer. The application layer communicates with the control layer via the northbound interface.

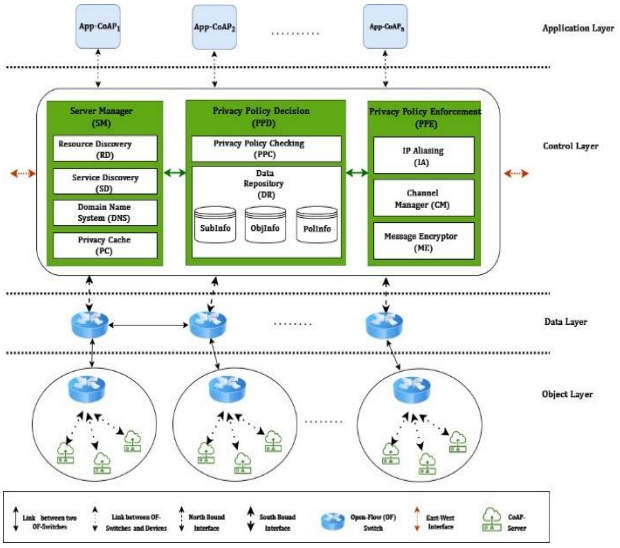

Fig. 1: Our proposed framework architecture.

Controller: The controller component is the most critical software-based network element, serving as the control center of the system. Its responsibilities include generating the internal switching paths of the network and managing network state change events. The controller in our prototype comprises three sub-components that implement different levels of privacy protection. When dealing with extensive or physically distributed systems, we add an additional set of controller nodes to the middleware layer to handle concurrent user requests. This approach ensures high load balancing and consistency while providing fast response times for a large number of requests. To communicate with these controller nodes, we use the West APIs if the system needs to be scaled. More details about the sub-components within this middleware are as follows:

Privacy Policy Decision (PPD): This component is responsible for storing information related to CoAP clients and CoAP services. It also stores policies that specify various levels of privacy for messages exchanged between clients and services. When a CoAP client requests access to a specific CoAP service, the component performs checks to determine the appropriate level of privacy. In summary, this component manages privacy policies for client and server CoAP communication based on their attributes.

The Privacy Policy Enforcement (PPE): It is responsible for enforcing the privacy protection decisions made by the Privacy Policy Decision (PPD) component. This is achieved by implementing privacy solutions that safeguard sensitive data and by employing mechanisms to conceal user identities. The Privacy Protection Enforcement (PPE) component accomplishes this through three subcomponents:

I. IP Aliasing (IA) Sub-Component: It allows a CoAP client to send a CoAP request to or receive a CoAP response from a CoAP server using a source IP address different from the one used in the original request. By employing IP aliasing, the CoAP client can maintain communication with the CoAP server even if the original IP address becomes unavailable.

II. Channel Manager (CM) Sub-Component: It allows the frequent change of the communication channel between two devices, namely the CoAP client and CoAP server. When a CoAP client sends a CoAP request to a CoAP server, the information included in the CoAP request is used by this sub-component to create a dynamic channel between the CoAP client and the CoAP server. For example, when a new packet (request) arrives at the controller and matches a certain flow in the flow table, the controller can use the packet counter (one of the flow table items) as a measure to determine whether a new channel is needed or the existing channel can be changed, used. A packet counter is a variable that counts the number of packets that match a particular flow. The controller can set a threshold for the packet counter and compare it to the current value of the counter for each flow. If the packet counter exceeds the threshold, the controller can create a new channel and add it to the channel table. Otherwise, the controller can update the existing channel with new packet information. In our work, this component is used for messages with sensitivity level 1.

III. Message Encryptor (ME) Sub-Component: It facilitates the encryption of CoAP message content, ensuring that only authorized entities can access the information. The ME sub-component employs various cryptographic algorithms to secure the payload of CoAP messages. In our study, this component is utilized for messages classified with sensitivity level 2.

Server Manager (SM) Component: It administers CoAP servers that deliver CoAP services to clients, facilitating more granular control over individual servers and enhancing monitoring and troubleshooting capabilities. Additionally, it optimizes the utilization of network resources, thereby reducing downtime and improving overall network performance. This component comprises four sub-components:

I. Resource Discovery (RD) Sub-Component: It is responsible for determining the location, ownership, and Uniform Resource Identifier (URI) of IoT resources across various domains.

II. Service Discovery (SD) Sub-Component: It maintains a registry of available CoAP services. When a CoAP client requires a specific CoAP service, it can submit a request to the SD sub-component to locate the appropriate CoAP service hosted on a CoAP server.

III. Domain Name System (DNS) Sub-Component: It maintains a mapping between resource URIs and their corresponding network locations. When a CoAP client sends a request to a specific CoAP server, the DNS sub-component resolves the URI in its mapping and returns the corresponding network location of the CoAP server.

IV. Privacy Cache (PC) Sub-Component: It is designed to store frequent responses from the PPD for CoAP services, thereby providing faster privacy-preserving services. Its primary responsibility is to enhance the overall performance of the proposed framework by reducing the time required to ensure privacy when accessing a specific CoAP service.

Data layer: This layer consists of OpenFlow switches that are interconnected by high-speed communication links and are responsible for packet forwarding. All the devices in this layer operate under the control of the software-based network controller. The communication between the data layer and the control layer occurs through the southbound interface, which is the standard OpenFlow interface in this case.

Object layer: This is the lowest layer in the proposed architecture, which comprises heterogeneous IoT devices (i.e., CoAP servers) that offers various services via CoAP protocol.

B. Proposed Attribute-Based Privacy Protection

The proposed framework employs an Attribute-Based Privacy Protection (ABPP) model to enforce varying levels of privacy protection during message transmission in Machine-to-Machine (M2M) Internet of Things (IoT) networks. This model comprises seven key components:

I. Actors (SUB): Individuals or entities performing actions.

II. Attributes of the Actors (SUB.ATT): Characteristics or properties of the actors.

III. Entities (OBJ): Objects or entities affected by the actions.

IV. Attributes of the Entities (OBJ.ATT): Characteristics or properties of the entities.

V. Contextual Conditions (ENV.ATT): Environmental attributes where the actions occur.

VI. Actions (OPS): The operations or activities performed.

VII. Policies (POL): Rules governing the application of different levels of privacy protection.

This structured approach ensures a comprehensive and adaptable privacy protection mechanism within M2M IoT networks.

Our proposed framework includes a Privacy Policy Decision (PPD) sub-component that evaluates the policies established by the Attribute-Based Privacy Protection (ABPP) model. In ABPP-SDN, we consider that CoAP messages can possess varying degrees of sensitivity. Initially, we assume three levels of sensitivity: Sensitive, Restricted, and Confidential. However, ABPP-SDN can be expanded to accommodate additional levels if necessary.

When a CoAP request, denoted as Req, is granted access to a CoAP service, the request is transmitted to the PPD component. The PPD component ensures the privacy of the CoAP response message based on the sensitivity of the original CoAP request. In our proposed ABPP model, every attribute related to subjects, objects, and the environment, as well as every operation, carries a degree of sensitivity. We assign a numerical value to represent the degree of sensitivity, where 0 signifies Sensitive, 1 denotes Restricted, and 2 indicates Confidential. To quantify the degree of sensitivity for them, we introduce a function called the Degree of Sensitivity (DoS), which measures their respective levels as follows:

$$A = DoS_{SUB.ATT_{Req}} = \frac{\sum_{l=1}^{n} DoS(SUB.ATT_l)}{2 \times n} \quad (1)$$

where *n* is the total number of subject's attributes in the request Req.

$$B = DoS_{OBJ.ATT_{Req}} = \frac{\sum_{l=1}^{m} DoS(OBJ.ATT_l)}{2 \times m} \quad (2)$$

where *m* is the total number of object's attributes in the request Req.

$$C = DoS_{ENV.ATT_{Req}} = \frac{\sum_{l=1}^{o} DoS(ENV.ATT_l)}{2 \times o} \quad (3)$$

where *o* is the total number of environment's attributes in the request Req.

$$D = \frac{DoS(OPS_{Req})}{2} \quad (4)$$

The degree of sensitivity of CoAP request can be calculated using the following equation:

$$DoS_{Req} = \frac{A + B + C + D}{4} \quad (5)$$

The PPE component can employ various techniques to protect the context of the CoAP response message based on the value of DoS_{Req},as shown in Table 2. The selection of these techniques is as follows:

Table 2: Privacy levels

$\mathbf{DoS_{Req}}$	Protection Solution	Praivacy Level
$\boldsymbol{DoS_{Req} \leq 0.35}$	*IP* Aliasing	0
$\mathbf{0.35 < DoS_{Req} < 0.65}$	Combination of IP Aliasin and Dynamic Channel	*1*
$\boldsymbol{DoS_{Req} \geq 0.65}$	Combination of IP Aliasin Dynamic Cannel and Encryption Content	2

- **Anonymity – IA sub-component:** When the DoS_Req value is less than or equal to 0.35, the IA sub-component employs the IP Aliasing technique to replace the original source IP address with a newly generated alias. The CoAP response is sent using this pseudonymous address, effectively concealing the identity of the client and ensuring source-level anonymity.
- **doUnLinkability – CM sub-component:** For 0.35 < DoS_Req < 0.65, the CM sub-component activates a combination of IP Aliasing and Dynamic Channel Switching. It evaluates flow-specific packet thresholds to select new transmission paths and alters communication routes unpredictably. This strategy anonymizes routing behavior and fragments traffic continuity—ensuring that incoming requests cannot be linked to outgoing responses, even under deep packet or flow pattern inspection by adversaries.
- **Untraceability – ME sub-component:** When DoS_Req ≥ 0.65, the ME sub-component implements a multi-layered privacy strategy comprising IP Aliasing, Dynamic Channel Switching, and Content Encryption. This integrated mechanism anonymizes the sender, obscures the transmission path, and secures message payloads against inspection. Consequently, adversaries are unable to trace the origin of messages or reconstruct communication flows—achieving robust untraceability throughout the transaction process.

PPD determines the appropriate level of privacy required for accessing a specific Constrained Application Protocol

(CoAP) service and subsequently communicates this privacy level to the Privacy Policy Enforcement (PPE) sub-component. Fig. 2 illustrates the workflow for ABPP enforcement within our proposed framework. As illustrated in Fig. 2, when a Constrained Application Protocol (CoAP) request (Req) is sent from the CoAP client to the Privacy Controller (PC), the getCache()

method is invoked to check whether the requested service has already been provided with privacy protection. If true, the PC sends the privacy level to the Privacy Policy Enforcement (PPE) sub-component; otherwise, it forwards the request to the Privacy Policy Decision (PPD) sub-component.

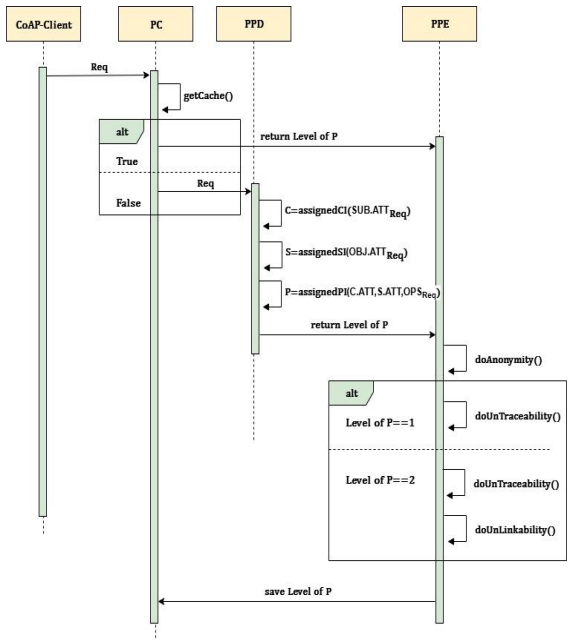

Fig. 2: Workflow for privacy protection enforcement.

The PPD sub-component maps the subject attribute of the request to a set of subjects by calling the assignedCI(SUB.ATTReq) method. All subjects and their attributes are then stored in the Context Information (CI) sub-component as an XML file. Subsequently, the PPD calls the assignedSI(OBJ.ATTReq) method to map the object attribute of the request onto a set of objects, which are stored in the Subject Information (SI) sub-component as an XML file.

The subject attribute (C), object attribute (S), and the requested operations (OPSReq) are mapped onto a set of policies using the assignedPI(C.ATT, S.ATT, OPSReq) method. These policies (POL) are defined by a system administrator and stored in the Policy Information (PI) sub-component as an XML file within our proposed framework.

The sub-components for anonymity, untraceability, and unlinkability execute the doAnonymity(), doUnTraceability(), and doUnLinkability() methods, respectively, based on the policy level received by the Privacy Policy Enforcement (PPE) sub-component. Finally, the PPE sends the privacy policy level to be saved in the Privacy Controller (PC) to reduce the time needed for privacy enforcement when accessing a specific CoAP service.

Algorithm 1 is designed to implement attribute-based privacy protection within our proposed framework. The input to the algorithm is a CoAP request, and it produces a secure environment as the output. The Privacy Controller (PC) component first checks whether the privacy level for the requested service is cached (lines 1-5). The Privacy Policy Decision (PPD) sub-component then retrieves the policy from the Context Information (CI), Subject Information (SI), and Policy Information (PI) sub-components that match the attribute set provided by the CoAP request (lines 6-8). Based on the policy, the level of privacy protection is determined and sent to the Privacy Policy Enforcement (PPE) sub-component (line 9). An anonymity technique is used by default at all privacy levels, which involves creating a new IP address to send CoAP request messages to a specific CoAP server (line 10). If the privacy level is one, only the untraceability technique is performed. Otherwise, the unlinkability technique is performed in addition to untraceability (lines 11-16). Finally, the PPE sends the privacy level to the PC to provide faster privacy-preserving services (line 17).

Algorithm 1. ABPP-SDN solution

Input: Req ← CoAP request {SUB.ATT, OBJ.ATT, OPSReq};
Output: Safe environment.

```
1:  if (getCache()) then
2:     return Level of privacy to PPE;
3:  else
4:     send Req to PPD;
5:  end If
6:  C← assignedCI(SUB.ATTReq);
7:  S← assignedSI(OBJ.ATTReq);
8:  P← assignedPI(C.ATT,S.ATT,OPSReq);
9:  getPL();//DoS
10: doAnonymity();// Algorithm 2
11:   if(p==1) then
12:         doUnLinkability ();// Algorithm 3
13:   else
14:         doUnTraceability ();// Algorithm 4
15:         doUnLinkability ();
16:   endif
17:  Send Level of Privacy to PC; the receiver applies
Algorithm 5 when untraceability is selected.
```

Algorithm 2 enhances privacy-preserving CoAP communication by integrating alias-based anonymization, flow-level session tracking, and cryptographically bound authentication. It begins (Line 1) by computing a unique FlowID from request attributes (SUB.ATT, OBJ.ATT, OPSReq) to distinguish sessions. To mitigate Sybil and spoofing attacks (Lines 2–4), the

client's identity is authenticated using its ID, IP address, and MAC address, and any requests originating from unregistered entities are denied. Alias assignment does as follows: if the flow is known (Lines 5–7), the system retrieves its existing alias IP; otherwise, a new pseudonymous IP is generated and stored (Lines 8–9), enabling per-flow anonymity. To guarantee alias integrity, an authentication tag (Auth_Tag) is constructed (Line 11) using HMAC over the alias IP, client IP, and flow ID. The CoAP request is then anonymized by replacing its source IP with the alias and embedding the tag (Lines 12–13). The alias IP is linked to the client in the internal registry (Line 14), and a flow rule is installed (Line 15) to rewrite source IPs in matching packets with time-bound enforcement. Session metadata—including alias, timestamp, and client ID—is logged for traceability (Line 16). Finally, the anonymized, authenticated request is forwarded securely to its destination (Line 17), completing the privacy-aware transmission cycle.

Algorithm 2. doAnonymity()

Input: Req ← CoAP request {SUB.ATT, OBJ.ATT, OPSReq}, Client_ID, Client_IP, MAC_Address;
Output: Anonymized and authenticated request with aliased source IP.

1. FlowID ← hash(SUB.ATT + OBJ.ATT + OPSReq)
2. if not isRegistered(Client_ID, Client_IP, MAC_Address) then
3. Reject Req
4. endif
5. if FlowID ∈ AliasTable then
6. Alias_IP ← AliasTable[FlowID]
7. else
8. Alias_IP ← GenerateNewAliasIP(Client_IP)
9. AliasTable[FlowID] ← Alias_IP
10. endif
11. Auth_Tag ← HMAC(Alias_IP ‖ Client_IP ‖ FlowID ‖ Secret_Key)
12. Req.sourceIP ← Alias_IP
13. Req.alias_tag ← Auth_Tag
14. IP_Alias_Table[Client_ID] ← Alias_IP
15. install_flow_rule(match: {FlowID, Client_ID, Client_IP}, action: rewrite_src_ip(Alias_IP),timeout: T)
16. Update Flow_Metadata_Table[FlowID] ← {Alias_IP, timestamp, Client_ID}
17. forward Req to next sub-com or destination

Algorithm 3, strengthens communication security by integrating session-aware flow recognition, replay defense, and dynamic channel switching. Initially (Line 1), a unique FlowInfo is derived from CoAP request parameters to distinguish sessions. To resist replay attacks (Lines 2–7), a timestamp and random nonce are embedded and validated per client; stale or duplicate requests are immediately rejected. The system then checks whether the flow is already tracked (Lines 8–14); if not, a new record is created and stored. As packets accumulate, a counter is incremented (Line 15) and evaluated against a rotation threshold (Lines 16–22). If exceeded, a new channel is randomly assigned to break traffic patterns; otherwise, the current channel is smoothly updated. The new channel is set (Line 23) and the request is routed through it (Line 24) before being forwarded to its destination (Line 25), completing a privacy-aware transmission cycle.

Algorithm 3. doUnLinkability ()

Input: Req ← CoAP request {SUB.ATT, OBJ.ATT, OPSReq}, Client_ID, Server_ID;
Output: randomized channel and replay protection.

1. FlowInfo ← hash(SUB.ATT + OBJ.ATT + OPSReq)
2. ts ← current_timestamp()
3. nonce ← generate_random_nonce()
4. Req.metadata ← {ts, nonce}
5. if not isFresh(ts, nonce, Client_ID) then
6. Reject Req
7. endif
8. if FlowInfo ∈ FlowTable then
9. flow ← FlowTable[FlowInfo]
10. else
12. flow ← CreateNewFlow(FlowInfo)
13. FlowTable[FlowInfo] ← flow
14. endif
15. flow.packetCounter ← flow.packetCounter + 1
16. if flow.packetCounter ≥ Threshold then
17. newChannel ← GenerateRandomChannel()
18. ChannelTable[FlowInfo] ← newChannel
19. flow.packetCounter ← 1
20. else
21. newChannel ← UpdateChannel(flow.currentChannel, flow.packetCounter)
22. endif
23. flow.currentChannel ← newChannel
24. Route Req via newChannel
25. forward Req

Algorithm 4 provides a robust mechanism for securing CoAP requests by integrating identity verification, replay protection, and sensitivity-aware encryption. The process begins by validating the legitimacy of the client through identity binding - ensuring that the Client_ID is correctly associated with the claimed Client_IP (Lines 1–3). This step mitigates Sybil and IP spoofing attacks, rejecting any request from unverified sources. To prevent replay attacks, the algorithm generates a

timestamp (ts) and a random nonce, which are validated for freshness (Lines 4–8). Requests that fail this freshness check are immediately discarded. The Algorithm 4, (Line 9), generates a metadata string based on core attributes of the request, name SUB.ATT, OBJ.ATT, and OPSReq, which capture contextual sensitivity regarding the operation. A Key Identifier (KID) is chosen corresponding to the requesting client‑server pair, and the associated Pseudorandom Key (PRK) is retrieved. The key in question was either a pre-shared key (PSK) or obtained through the Elliptic Curve Diffie–Hellman (ECDH) process (Lines 10–12). A session key is generated from the PRK through the HMAC-based Key Derivation Function (HKDF), using the nonce as salt and a context string comprising metadata, algorithm ID, and KID (Line 13). This guarantees that the encryption key is context-aware and unique. So, in order to protect the payload, the algorithm chooses a suitable Authenticated Encryption with Associated Data (AEAD) scheme like AES-CCM or ChaCha20-Poly1305, which varies from one to the other according to the sensitivity of the metadata (Line 14). Thereafter, it takes the payload and encrypts it with the session key and generates the cryptographic tag for ensuring integrity and authenticity (Line 15). The AAD consists of the timestamp, nonce, and flow identifier, binding the encryption to the request context.

Algorithm 4. doUntraceability ()

Input: Req ← CoAP request {SUB.ATT, OBJ.ATT, OPSReq}, Client_ID, Client_IP;
Output: Encrypted and integrity-protected request.

1. if not isBound(Client_ID, Client_IP) then
2. Reject Req
3. endif
4. ts ← current_timestamp()
5. nonce ← generate_random_nonce()
6. if not isFresh(ts, nonce, Client_ID) then
7. Reject Req
8. endif
9. Metadata ← (SUB.ATT ‖ OBJ.ATT ‖ OPSReq)
10. KID ← select_key_id(Client_ID, Server_ID)
11. PRK ← get_secret(KID) // pre-shared key or ECDH output
12. Context ← (Metadata ‖ AlgorithmID ‖ KID)
13. SessionKey ← HKDF(PRK, salt=nonce, info=Context)
14. EncryptionAlgorithm ← ChooseBasedOnSensitivity(Metadata)// AES-CCM or ChaCha20-Poly1305
15. (Ciphertext, Tag) ← AEAD_Encrypt(Req.payload, SessionKey, AAD={ts, nonce, FlowID})
16. Req.payload ← Ciphertext
17. Req.header ← Req.header ∪ {AlgorithmID, KID, NonceID: nonce, ts, Tag}
18. forward Req

Finally, the encrypted payload replaces the original content, and the request header is augmented with security metadata including the algorithm ID, key identifier, nonce reference, timestamp, and authentication tag (Lines 16–17). The secured request is then forwarded for processing (Line 18).

Algorithm 5 describes the reception of a decrypted payload or an offense. The receiver first checks freshness against the timestamp and nonce; if it fails the replay protection check, the request is rejected right away (Lines 1-3). Metadata are framed from core CoAP attributes SUB.ATT, OBJ.ATT, and OPSReq to signify the context of the original request (Line 4). The receiver uses the KID to retrieve the PRK. This must either be a PSK or be derived from an ECDH exchange to remain consistent with the sender (Line 5). A session key is generated using the HKDF with nonce as salt and a context string containing metadata, algorithm ID, and KID (Lines 6-7). With this, AEAD is leveraged to decrypt the encrypted payload. The AAD includes the timestamp, nonce, and flow identifier, and the received authentication tag is used to verify successful decryption (Line 8). If decryption or validation fails, the request is rejected to preclude any possibility of distinguishing the request or forgery (Line 9). Once the message has been successfully decrypted, the plaintext payload will replace the encrypted payload, and a CoAP request containing the payload will be sent to the CoAP service for processing (Lines 10-11).

Algorithm 5. Receiver: doDecryption

Input: Req ← CoAP request with {AlgorithmID, KID, nonce, ts, Tag}, Metadata=(SUB.ATT, OBJ.ATT, OPSReq);
Output: Decrypted payload or Reject.

1: if (not isFresh(ts, nonce, Client_ID)) then
2: **Reject**;
3: end if
4: Metadata ← (SUB.ATT ‖ OBJ.ATT ‖ OPSReq)
5: PRK ← get_secret(KID) // same PSK or ECDH context as sender
6: Context ← (Metadata ‖ AlgorithmID ‖ KID)
7: SessionKey ← HKDF(PRK, salt=nonce, info=Context)
8: Plaintext ← AEAD_Decrypt(Req.payload, SessionKey, AAD={ts, nonce, FlowID}, tag=Tag)
9: if decryption/authentication fails then Reject
10: Req.payload ← Plaintext
11: deliver Req to CoAP service

Results and Discussion

In this section, we first describe our simulation topology and its associated settings, followed by a presentation and analysis of the experimental results,

performance analysis and computational overhead evaluation and security analysis.

A. *Experimental Settings*

We used the CloudSimSDN [35] to simulate the proposed framework.

All the experiments were run PC with an Intel Core i5-8265U CPU @ 1.8GHz, 16 GB RAM, running Microsoft Windows 10 64-bit.

We configured the IoT network using the physicalTopologyGenerator class in CloudSimSDN, which enabled us to save and load the network topology in JSON file format.

The network topology consisted of one controller (represented by the Network Operating System class in CloudSimSDN, responsible for managing the overall network behavior of the simulation), four OpenFlow switches (SW1, SW2, SW3, and SW4), 50 CoAP clients, 30 CoAP servers, and 80 virtual machines (VM1, VM2, ..., VM80). The VMs were deployed on physical nodes, and network packets were routed between nodes via OpenFlow switches. Additionally, we set the network latency to 0.1 milliseconds and the network bandwidth to 250 Mbps.

Fig. 3 illustrates the network topology used in our experiments.

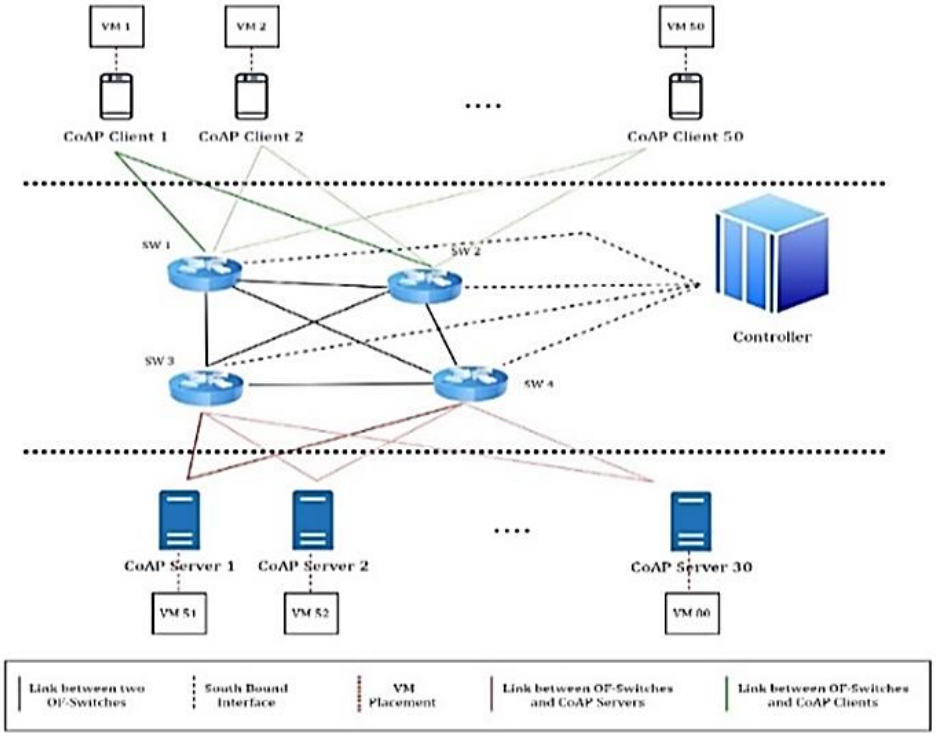

Fig. 3: Network topology used in experiments.

B. *Experimental Results*

Fig. 4 illustrates the average response time for protecting the privacy of CoAP messages in our proposed framework, with the number of CoAP requests varying from 100 to 1000 in increments of 100. Fig. 4 clearly demonstrates that employing two or three privacy protection techniques simultaneously does not significantly increase the average response time. However, it also shows that the average response time is higher when all three techniques—IP Aliasing (doAnonymity()), Dynamic Channel (doUnTraceability()), and Content Encryption (doUnLinkability())—are used together, compared to when only one or two techniques are employed for privacy protection in the ABPP model.

To further analyze the results, we conducted simulations with varying numbers of CoAP requests and applied different combinations of privacy protection techniques.

The average response time for 1000 requests is as indicate that while the use of multiple privacy protection techniques increases the average response time, the increase is not substantial.

Moreover, incorporating a cache into the proposed framework reduces response time by eliminating the need to execute new operations for every request, highlighting the framework's efficiency and effectiveness in protecting CoAP message privacy.

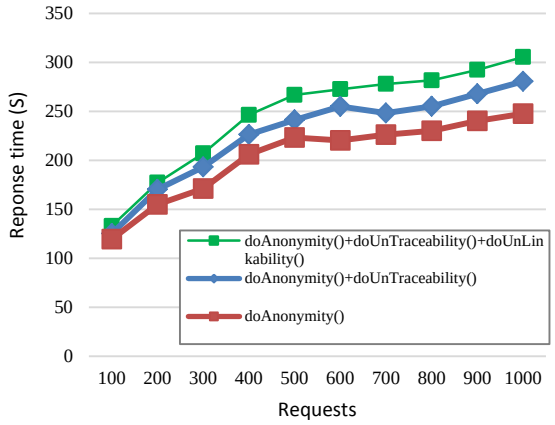

Fig. 4: Average response time for privacy protection.

Fig. 5 presents the average throughput for protecting the privacy of CoAP messages within our proposed framework, with the number of CoAP requests varying. As depicted in Fig. 5, it is evident that employing additional privacy protection techniques for CoAP requests results in a decrease in average throughput. This inverse relationship between throughput and response time indicates that the average throughput diminishes further when all three techniques are applied simultaneously.

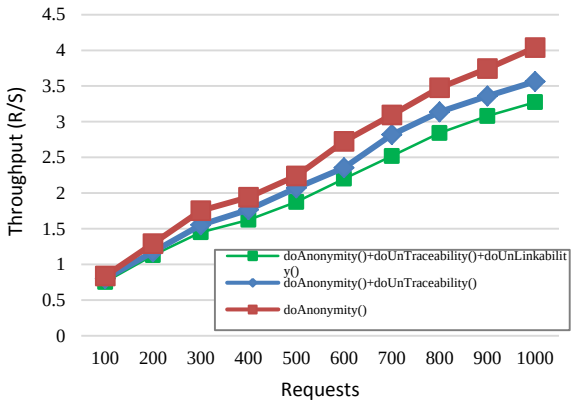

Fig. 5: Average throughput for privacy protection.

Fig. 6 illustrates the CPU utilization for privacy protection within our proposed framework, with varying numbers of CoAP requests. As demonstrated in Fig. 6, the CPU utilization is higher when the ABPP model employs the combination of IP Aliasing, Dynamic Channel, and Content Encryption compared to other scenarios.

This increased utilization is attributed to the greater computational resources required for privacy protection operations, such as generating new IP addresses, switching communication channels, and encrypting content.

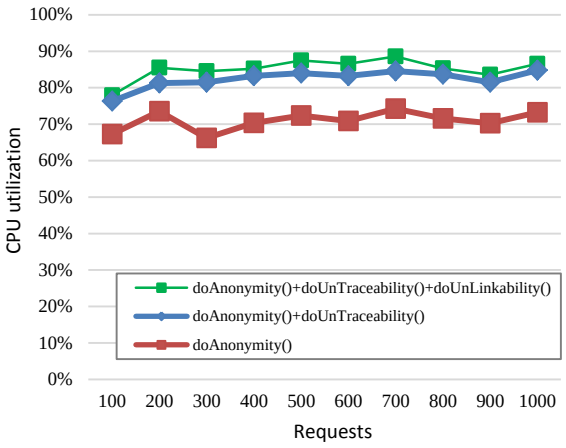

Fig. 6: CPU utilization for privacy protection.

C. *Performance Analysis and Computational Overhead Evaluation*

One of the limitations of the proposed framework is the computational overhead introduced by the layered privacy mechanisms in ABPP-SDN. To mitigate this issue, a caching mechanism was employed. A series of simulations was conducted using the CloudSimSDN environment to evaluate performance. The results, presented in Fig. 7 and Fig. 8, provide a comparative analysis of performance metrics across different configurations: ABPP-SDN with caching (featuring PC capability), ABPP-SDN without caching, and baseline methods from previous studies [27] and [30].

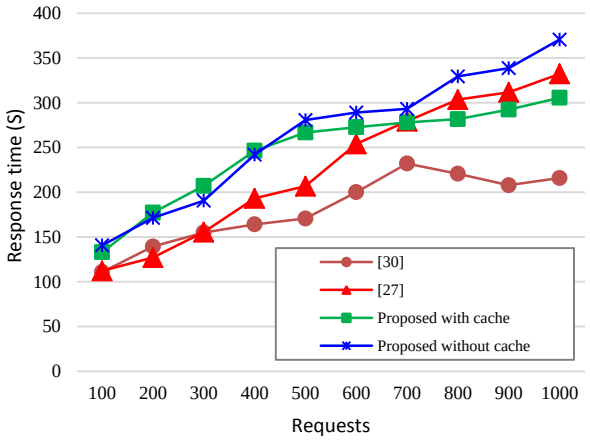

Fig. 7: Average response time.

The integration of the storage mechanism (PC) within the ABPP-SDN architecture demonstrably mitigates computational overhead, resulting in reduced response latency and substantially lower CPU usage. As depicted in Fig. 8, CPU consumption exhibits a direct correlation with the volume of CoAP requests; however, the variant of ABPP-SDN enhanced with caching consistently

maintains lower resource utilization compared to its non-caching counterpart. This observation underscores the computational efficiency of the proposed framework and reinforces its viability for deployment in resource-constrained IoT platforms.

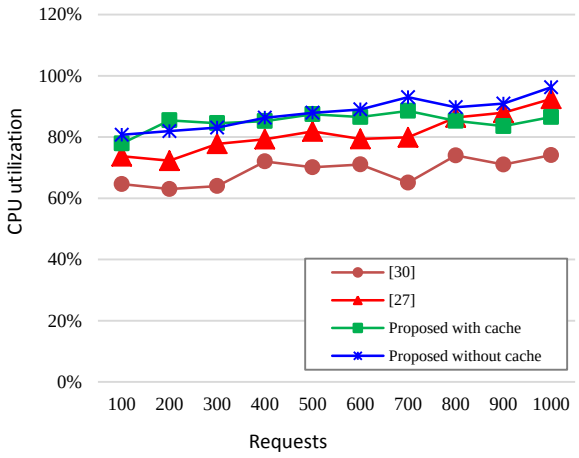

Fig. 8: CPU utilization.

D. *Energy Consumption Analysis*

To evaluate the framework's suitability for resource-constrained IoT environments, we conducted targeted simulations using CloudSimSDN to measure energy consumption under varying CoAP request volumes. The results indicate that the ABPP-SDN architecture equipped with the caching mechanism (PC) exhibits notably lower energy usage compared to its non-caching counterpart and [27], [30]. This reduction stems from the avoidance of repeated computations and the reuse of previously coordinated responses, which minimizes the frequency and intensity of processing and communication tasks.

The energy-aware behavior of the caching-enabled model confirms the framework's capacity to maintain privacy protection while optimizing resource consumption, validating its efficiency for deployment in low-power IoT platforms.

Fig. 9 illustrates the energy consumption behavior of the proposed ABPP-SDN framework under varying CoAP traffic volumes, comparing its caching-enabled and non-caching configurations and [27], [30].

As shown, the caching mechanism significantly reduces energy usage by minimizing repetitive processing and network activities. The energy consumption of the proposed with cache is on average 27.5% better than [27] and 30.4% more efficient than [30].

This validates the framework's resource-awareness and confirms its suitability for deployment in energy-constrained IoT infrastructures.

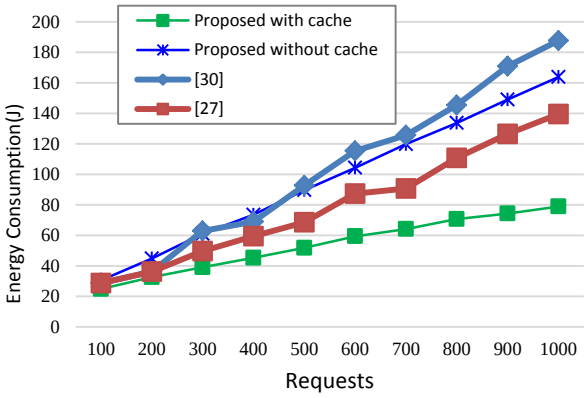

Fig. 9: Energy consumption.

E. *Packet Delivery Ratio (PDR)*

To evaluate the performance of the proposed framework in terms of Packet Delivery Ratio (PDR), a simulation was conducted using the CloudSim-SDN environment.

At a specific point during the simulation, a random path failure was introduced by disabling one of the switches or links in the network. This disruption was designed to emulate unpredictable failures in real-world IoT environments. The PDR was calculated using the following formula:

$$PDR = \frac{Total\ Packets\ Received}{Total\ Packets\ Sent} * 100 \quad (6)$$

This scenario was executed multiple times under varying traffic loads and topologies to obtain an average PDR for each method under comparison.

The results of the simulation revealed that our proposed framework, the ABPP model, achieved superior packet delivery performance compared to existing approaches. Table 3 summarizes the average PDR values.

Table 3: PDR

Method	Average PDR (%)
ABPP with cache	94.3
ABPP without cache	90.1
[27]	89.2
[30]	86.7

The results indicate that the combination of ABPP, encryption techniques, and adaptive routing mechanisms in our framework markedly reduces packet loss while enhancing reliability, especially under the high-traffic and failure-prone conditions typical of smart city IoT systems

F. Throughput Analysis

Throughput is a key performance metric that quantifies the amount of data successfully delivered over a network per unit time. It reflects the efficiency of data transmission and is particularly critical in resource-constrained IoT environments. The throughput is calculated using the following formula:

$$Throughput = \frac{\sum_{j=1}^{P_{received}} s_j}{T} \quad (7)$$

s_j: $Size\ of\ the\ j^{th}\ successfully\ received\ packet\ (in\ bits)$
$P_{received}$: $Total\ number\ of\ successfully\ received\ packets$
T: $Total\ simulation\ or\ measurement\ time\ (in\ seconds)$

Units of throughput are typically expressed in bits per second (bps), kilobits per second (Kbps), or megabits per second (Mbps).

To validate the performance of the proposed framework, throughput results are compared against [27], [30]. The comparative data is summarized in Table 4.

Table 4: Average throughput

Method	Average Throughput (Kbps)
ABPP with cache	512.4
ABPP without cache	476.3
[27]	472.1
[30]	438.7

Indeed, the performance in terms of throughput for the implemented ABPP-SDN framework has been found to be more potent than that of the current available privacy-preserving techniques as reflected in the simulation results. Multiple privacy-enhancing mechanisms, from anonymity and unlinkability to untraceability, IP aliasing, dynamic channel switching, and payload encryption, are well placed within the framework while delivering a very high data delivery rate.

Thus, the system can protect critical information about clients without compromising upon transmission efficiency. Decoupling message sensitivity levels from privacy requirements, ABPP-SDN is able to better optimize the routing quality through SDN, thus achieving a more optimal security-performance trade-off for smart-city IoT environments, where both privacy and responsiveness are critical.

G. Security Analysis

In this section, we present a comprehensive security evaluation of the proposed framework, encompassing both formal verification and informal analysis techniques.

- Informal Analysis

Anonymity: The proposed framework enforces anonymity by decoupling a client's true identity ID_C and real IP address IP_C in transmitted packets via IP aliasing, denoted as $IP_A = F_{alias}(ID_C, IP_C)$. To prevent spoofing, each alias is bound to an authenticated mapping and verified using an authentication tag $Tag_A = HMAC_K(IP_A \parallel IP_C)$, where $HMAC$ is a hash-based message authentication code and K is a controller-issued secret key. This binding ensures that even if an attacker forges IP'_C, the mismatch with Tag_A will result in rejection.

Unlinkability: It is achieved via dynamic channel switching, where each flow f_i between a client and server is assigned a per-session channel $CH_i \in \mathcal{C}$, updated when a threshold condition $Counter_i > \theta$ is met. This prevents correlation of successive requests $R_i, R_{i+1}, \ldots$ from being linked through static flow or path patterns, thus concealing user activity over time.

Untraceability and man-in-the-middle (MITM) attacks: Untraceability of content is guaranteed by encrypting CoAP payloads m as $Enc_K(m)$, with optional encryption of metadata based on sensitivity level $\sigma \in \{0,1,2\}$. Furthermore, each message includes a MAC tag $MAC = HMAC_K(m \parallel ts \parallel nonce)$ to ensure integrity and prevent message tampering, effectively mitigating MITM attacks.

Replay attacks: To counter these attacks, the controller and CoAP server maintain a replay window $\mathcal{W}_{ID_C} \subset \mathbb{T} \times \mathbb{N}$, where $\mathbb{T}$ is the timestamp space and $\mathbb{N}$ is the nonce space. A message with $(ts, nonce) \in \mathcal{W}$ is considered invalid. Assuming secure clocks and bounded drift ΔT, this defense remains efficient and lightweight.

Sybil attacks: To mitigate sybil attacks, each client is required to register through a validation mechanism that binds $ID_C \leftrightarrow IP_C \leftrightarrow MAC_C$. Multiple identity claims from a single MAC or IP subnet are detected through statistical thresholds $N_{IDs}(IP_C) > \delta$, where δ is a system-defined sybil detection parameter.

IP spoofing: It is neutralized via source validation by the SDN controller, which drops packets where $IP_{src} \notin \mathcal{M}_{ID_C}$, with $\mathcal{M}$ being the maintained mapping of legitimate identities to IP/MAC pairs.

Others security concerns: The system is resilient to traffic analysis attacks by leveraging per-flow encryption and dynamic alias rotation, thereby disrupting pattern matching. It is also resistant to resource exhaustion or Denial of Service (DoS) attacks through rate-limiting policies: any client ID_C sending more than λ messages in

a time window Δt will be temporarily throttled or quarantined.

- Cryptographic Instantiation and Decryption Process

For CoAP messages assigned the untraceability level, the payload is encrypted with a session key derived via HKDF-SHA256. The secret input, Pseudorandom Key (PRK), is either a pre-shared key (PSK) or an ephemeral Diffie–Hellman output, referenced by a short Key Identifier (KID). The nonce, generated per message, is included in the header and used as HKDF salt, ensuring freshness and binding keys to individual transmissions. The HKDF info string is defined as (SUB.ATT ‖ OBJ.ATT ‖ OPSReq ‖ AlgorithmID ‖ KID).

The resulting session key is used with an Authenticated Encryption with Associated Data (AEAD) scheme. By default, we employ AES-CCM-128, the recommended mode in constrained IoT and OSCORE, though ChaCha20-Poly1305 and AES-GCM are also supported. Integrity and replay protection are achieved by: (i) authenticating the (timestamp, nonce, FlowID) fields as AEAD associated data (AAD), and (ii) checking freshness of (timestamp, nonce) at the receiver. Keys are never transmitted; both sender and receiver compute them independently.

- Security proof using ROM

The Random Oracle Model (ROM) is a theoretical framework in which all parties (including attackers) interact with a public oracle $\mathcal{O}_H$ that responds to each unique input x with a truly random output $y = \mathcal{O}_H(x)$, consistent across repeated queries [36]. In ROM, cryptographic hash functions (e.g., SHA-256) are treated as idealized random functions. Let's define a series of games and evaluate the attacker's advantage $Adv_{\mathcal{A}}$ under ROM. We assume a probabilistic polynomial-time attacker $\mathcal{A}$, and define the following standard security games:

Anonymity Game ($\mathbb{G}_{anon}$). In this game, the adversary tries to distinguish between two clients ID_0 and ID_1 sending anonymized messages via IP aliasing. Challenger picks $b \xleftarrow{\$} \{0,1\}$, uses ID_b to generate a message with alias IP_A to $\mathcal{A}$ who outputs guess b'. We assume IP aliasing is randomized per client via $IP_A = \mathcal{O}_H(ID \parallel nonce)$ with inaccesible mapping table. The advantage of $\mathcal{A}$ is:

$$Adv_{\mathcal{A}}^{anon} = \left|\Pr[\mathcal{A} \to b'] - \frac{1}{2}\right| \leq \epsilon \quad (8)$$

Under ROM, $\mathcal{A}$'s probability of linking alias IP_A to a specific ID is negligible unless it breaks the oracle (which behaves randomly).

Unlinkability Game ($\mathbb{G}_{unlink}$). In this game, the adversary determines whether two messages come from the same user despite dynamic channel switching and aliasing. The challenger prepares two messages m_0 and m_1: one from a repeated session (same user) and one from a new user (both encrypted via different pseudonymous channels). $\mathcal{A}$ receives both and guesses which is the repeat session. We assume dynamic channels $CH_i = \mathcal{O}_H(ID \parallel session_{nonce})$ and IPs and flows are randomized per session. $\mathcal{A}$'s advantage is:

$$Adv_{\mathcal{A}}^{unlink} = \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \epsilon \quad (9)$$

$\mathcal{A}$ has negligible advantage since all observable values (IP, channel ID, message tags) are randomized through the oracle.

Message Privacy Game ($\mathbb{G}_{priv}$). Here, the adversary distinguishes between two encrypted messages m_0 and m_1 under symmetric encryption with ROM-derived keys. The challenger picks bit $b \in \{0,1\}$, encrypts m_b using key $K = \mathcal{O}_H(ID_C \parallel ID_S)$ and transmits ciphertext $c = Enc_K(m_b)$ to $\mathcal{A}$. $\mathcal{A}$ receives ciphertext c, tries to guess b. We assume encryption is secure against indistinguishability under chosen plaintext attacks (IND-CPA) and keys derived via ROM hash oracle. The advantage of $\mathcal{A}$ in winning the game is:

$$Adv_{\mathcal{A}}^{priv} \leq \epsilon_{enc} + qH \cdot 2^{-n} \quad (10)$$

where, ϵ_{enc} is the IND-CPA advantage, qH is the number of hash queries, and n is hash output length (256 bits). This shows that the attacker can only win if it queries the oracle with the correct input used to derive K, which is infeasible for large n.

Conclusion

In this paper, we presented a novel four-layer SDN-based framework designed to enhance privacy protection in IoT networks. The framework utilizes an ABPP model to specify policies and employs various privacy protection techniques, such as IP aliasing, dynamic routing, and content encryption, to enhance the privacy of CoAP messages during transmission based on their sensitivity levels.

The framework anonymizes all CoAP messages using IP aliasing and applies dynamic routing for sensitive CoAP messages. Additionally, it employs different encryption algorithms for highly sensitive CoAP messages, augmenting existing techniques to prevent data disclosure within CoAP messages. We implemented the proposed framework using CloudSimSDN and evaluated its performance through several experiments. The use of a cache in our proposed framework helps reduce the response time by avoiding the need to perform new operations for each request. This demonstrates the efficiency and effectiveness of our framework in providing privacy protection for CoAP messages.

Although the proposed framework exhibits higher response times compared to existing approaches from [30], that use only one method for privacy preservation, this limitation is compensated by significant improvements in other critical performance metrics. Basically, the incorporation of dynamic routing and a cache mechanism into an SDN-based architecture brings out improved energy efficiency, higher rates of packet delivery, and better overall throughput. These optimizations become critical in resource-constrained IoT environments, where efficiency gains of the network and its scaling capability are concerned. Therefore, it reflects real benefits in operational up-hold by our design in return for the latency penalty being incurred.

The incorporation of dynamic routing and cache mechanism to the SDN-based architecture has specifically yielded gains in more efficient energy consumption, increased packet delivery rates, and better overall throughput. Such optimizations become critical in resource-constrained IoT environments, for efficiency gains and scalability of the network.

Thus, the latency trade-off is well warranted with the broader operational advantages gained through our design.

We provided a security proof using random oracle model and informal analysis shows that the proposed framework is secure against anonymity violation attacks, MITM attacks, replay attacks, sybil attacks, and IP spoofing.

In the future, we plan to extend this research in two directions:

- Extending our proposed ABPP model to explore the integration of machine learning algorithms to dynamically adjust privacy-preserving techniques based on real-time network conditions and threat levels.
- Developing novel privacy prevention techniques to avoid network traffic analysis in our proposed framework.
- Re-architecting the proposed framework for edge processing to further improve computational overhead.

Author Contributions

Conceptualization and design: S.H. Erfani; Formal Analysis: A. Sahafi; Software: Sh. Zangaraki; Research: three authors; Writing - preparation of the first draft: Sh. Zangaraki; Writing - review and editing: S.H. Erfani; Supervision: S.H. Erfani and A. Sahafi.

Acknowledgment

The authors express their sincere gratitude to Dr. Meghdad Mirabi for his invaluable support and expert guidance in formulating the research problem.

Funding

This research received no external funding.

Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

Abbreviations

SDN	Software Define Network
ABPP	Attribute Base Privacy Preserving
PPD	Privacy Policy Decision
PPE	Privacy Policy Enforcement
IA	IP Aliasing
CM	Channel Manager
ME	Message Encryptor
SM	Server Manager
RD	Resource Discovery
SD	Service Discovery
DNS	Domain Name System
PC	Privacy Cache

References

[1] 20 10 2023. [Online]. Available:https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[2] D. E. Kouicem, A. Bouabdallah, H. Lakhlef, “Internet of things security: A top-down survey,” Comput. Networks, 141: 199-221, 2018.

[3] G. Yang, "An Overview Of Current Solutions For Privacy In The Internet Of Things," Front. Artif. Intell., 5(812732): 8, 2022.

[4] H. Ahmadvand, C. Lal, H. Hemmati, M. Sookhak, M. Conti, “Privacy-preserving and security in sdn-based IOT: A survey,” IEEE Access, 11: 44772 - 44786, 2023.

[5] M. Amiri-Zarandi, R. A. Dara, E. Fraser, "A survey of machine learning-based solutions to protect privacy in the Internet of Things," Comput. Secur., 96, 101921, 2020.

[6] B. Ayodele, V. Buttigieg, “SDN as a defence mechanism: a comprehensive survey,” Int. J. Inf. Secur., 23(1): 141-185, 2024.

[7] S. Bera, S. Misra, A. . V. Vasilakos, "Software-defined networking for internet of things: A survey," IEEE Internet Things J. , 4(6): 1994-2008, 2017.

[8] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. Esteve Rothenberg, S. Azodolmolky, "Software-defined networking: A comprehensive survey," Proc. IEEE, 103(1): 14-76, 2015.

[9] P. M. Chanal, M. S. Kakkasageri, “Security and privacy in IoT: A survey,” Wireless Pers. Commun., 115(2): 1667-1693, 2020.

[10] S. Ullah, S. Ullah, M. Imran, “Recent developments in authentication schemes used in machine-type communication

devices in machine-to-machine communication: Issues and challenges," Comput. Mater. Continua, 79(1): 93-115, 2024.

[11] S. R. Mishra, B. Shanmugam, K. C. Yeo, S. Thennadil, "SDN-enabled IoT security frameworks—A review of existing challenges," Technologies, 13(3): 121, 2025.

[12] C. Bormann, A. P. Castellani, Z. Shelby, "CoAP: An application protocol for billions of tiny internet nodes," IEEE Comput. Soc., 16(2): 62-67, 2012.

[13] Z. Shelby, K. Hartke, C. Bormann, "The Constrained Application Protocol (CoAP)," Internet Eng. Task Force (IETF), 2014. [Online]. Available: https://tools.ietf.org/html/rfc7252.

[14] W. Bekri, R. Jmal, L. C. Fourati, "Internet of things management based on software defined," Int. J. Wireless Inf. Networks, 27(September): 385-410, 2020.

[15] H. Farhady, H. Lee, A. Nakao, "Software-defined networking: A survey," Comput. Networks, 81: 79-95, 2015.

[16] I. Alsmadi, D. Xu, "Security of software defined networks: A survey," Comput. Secur., 1(53): 79-108, 2015.

[17] B. MISHRA, A. KERTESZ, "The use of MQTT in M2M and IoT Systems: A Survey," IEEE Access, 8: 201071-201086, 2020.

[18] W. Wang, H. Zhao, J. Zhu, "GRPC: A communication cooperation mechanism in distributed systems," ACM SIGOPS Oper. Syst. Rev., 27(3): 75–86, 1993.

[19] S. Zangarak, M. Mirabi, S. H. Erfani, A. Sahafi, "SecShield: An IoT access control framework with edge caching using software defined network," Peer-to-Peer Networking Appl., 18(56), 2025.

[20] M. Seliem, K. Elgazzar, K. Khalil, "Towards privacy preserving iot environments: A survey," Wireless Commun. Mobile Comput., 2018(2): 15, 2018.

[21] A. A. A. Sen, F. A. Eassa, K. Jambi, M. Yamin , "Preserving privacy in internet of things: a survey," Int. J. Inf. Technol., 10(2): 189-200, 2018.

[22] G. Yang, "An overview of current solutions for privacy in the internet of things," Front. Artif. Intell., 5, 812732: 8, 2022.

[23] Q. Razi, R. Piyush, A. Chakrabarti, A. Singh, V. Hassija, G. S. Chalapathi, "Enhancing data privacy: A comprehensive survey of privacy-enabling technologies," IEEE Access, 13: 40354 – 40385, 2025.

[24] M. Gheisariy, G. Wang, W. Z. Khanz, C. Fernandez-Campusano, "A context-aware privacy-preserving method for IoT-based smart city using software defined networking," Comput. Secur., 87, 101470, 2019.

[25] V. Sridharan, K. Sudheera, K. Liyanage, M. Gurusamy, "Privacy-aware switch-controller mapping in SDN-based IoT networks," in Proc. International Conference on COMmunication Systems & NETworkS (COMSNETS), 2020.

[26] M. Gheisari, G. Wang, S. Chen, "An edge computing-enhanced internet of things framework for privacy-preserving in smart city," Comput. Electr. Eng., 81: 106504, 2020.

[27] M. Gheisari, G. Wang, S. Chen, H. Ghorbani, "IoT-SDNPP: A method for privacy-preserving in smart city with software defined networking," in Proc. International Conference on Algorithms and Architectures for Parallel Processing, 2018.

[28] M. Gheisari, G. Wang, S. Chen, A. Seyfollahi, "A method for privacy-preserving in IoT-SDN integration environment," in Proc. IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), 2018.

[29] C. Ke, Z. Zhu, F. Xiao, Z. Huang, Y. Meng, "SDN-based privacy and functional authentication scheme for fog nodes of smart healthcare," IEEE Internet Things J., 9(18): 17989-18001, 2022.

[30] J. A. Alzubi, A. Movassagh, M. Gheisari, H. E. Najafabadi, A. A. Abbasi, Y. Liu, Z. Pingmei, M. Izadpanahkakhk, A. P. Najafabadi, "A dynamic SDN-based privacy-preserving approach for smart city using trust technique," in Proc. 2022 9th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS): 1-5, 2022.

[31] S. R. Alotaibi, H. Alfraihi, N. Alruwais, M. Maray, A. B. Miled, A. M. Al-Sharafi, M. Alotaibi, S. H. Alajmani, "Two-tiered privacy preserving framework for software-defined networking driven defence mechanism for consumer platforms," IEEE Access, 13: 26684 – 26694, 2025.

[32] M. Gheisari, H. Tahaei, M. Malik, E. Mnkandla, Z. Wang, "A flexible software-defined networking-based privacy-preserving method for internet of things-based smart city environment based on the neighbors situation," Comput., 58(5): 27-36, 2025.

[33] M. Conti, P. Kaliyar, C. Lal, "CENSOR: Cloud-enabled secure IoT architecture over SDN," Concurrency Comput. Pract. Exper., 31(8): e4978, 2019.

[34] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, A. Rindos, "SDIOT: a software defined based internet of things framework," J. Ambient Intell. Human Comput., 6(4): 453-461, 2015.

[35] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, Y. Y. Buyya, "CloudSimSDN: Modeling and simulation of software-defined cloud data centers," in Proc. 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2015.

[36] S. ZakeriKia, R. Hajian, S. H. Erfani, A. M. Rahmani, "Robust and anonymous handover authentication scheme without key escrow problem in vehicular sensor networks," Wireless Networks, 27(7): 4997-5028, 2021.

Biographies

Shahrbanoo Zangaraki was born in Farahan, Iran, in 1984. She received the B.Sc. in Software Engineering from Arak University, Iran in 2007, the M.Sc. in Software Engineering from Islamic Azad University of Arak, Iran, in 2009. She is currently pursuing the full-time Ph.D. degree in Computer Engineering in Islamic Azad University of South Tehran Branch, Iran. She joined the Department of Computer Engineering, Islamic Azad University of Khomein, as an instructor. Her research interests include SDN, internet of things, security and privacy.

- Email: sh.zangaraki@gmail.com
- ORCID: 0000-0002-1931-4865
- Web of Science Researcher ID: AAO-5817-2021
- Scopus Author ID: 35811598900
- Homepage: NA

Seyed Hossein Erfani received his B.Sc. in Computer Engineering from Azad University, North Tehran branch, Iran, in 2006, the MS and PhD degrees in Computer Engineering from Azad University, Science and Research branch, Iran, in 2009 and 2014. Currently, he is an Associate Professor in the department of Computer Engineering at Azad University, South Tehran branch, Iran. He is the author/co-author of more than 10 publications in technical journals and conferences. His research interests are in the areas of security, cloud computing, wireless networks, internet of things and evolutionary computing.

- Email: h_erfani@azad.ac.ir
- ORCID: 0000-0002-7893-4191
- Web of Science Researcher ID: NA
- Scopus Author ID: 25421134100
- Homepage: NA

Amir Sahafi received the B.Sc. degree from Shahed University of Tehran, Iran, in 2005, and the M.Sc. and Ph.D. degrees from Islamic Azad University, Science and Research Branch, Tehran, in 2007 and 2012, respectively, all in Computer Engineering. He is currently an Assistant Professor with the Department of Computer Engineering, Islamic Azad University, South Tehran Branch, Tehran. His current research interests include distributed and cloud computing.

- Email: sahafi@iau.ac.ir
- ORCID: 0000-0002-6555-670X
- Web of Science Researcher ID: NA
- Scopus Author ID: 24528878600
- Homepage: NA