# A Comparison of Authentication Protocols for Multi-access Edge Computing

**Felipe Ponsano Franke**
Center of Mathematics, Computing and Cognition,
Federal University of ABC, Santo André, SP, Brazil
*felipe.franke@aluno.ufabc.edu.br*

and

**Carlo Kleber da Silva Rodrigues**
Center of Mathematics, Computing and Cognition,
Federal University of ABC, Santo André, SP, Brazil
*carlo.kleber@ufabc.edu.br*

and

**Rodrigo Augusto Cardoso da Silva**
Center of Mathematics, Computing and Cognition,
Federal University of ABC, Santo André, SP, Brazil
*cardoso.rodrigo@ufabc.edu.br*

**Abstract**

Multi-access Edge Computing (MEC) extends cloud computing capabilities to the network edge to reduce latency and enhance user experience. However, the constrained resources of edge devices and their exposure to cyber attacks make secure and efficient authentication a critical challenge. Although numerous authentication protocols have been proposed for MEC, the absence of standardized evaluation frameworks has hindered systematic comparison among existing solutions. Based on prior literature, this paper presents a comparative analysis of representative authentication protocols for MEC in an organized manner, evaluating and contrasting them according to their cryptographic foundations, computational and communication overhead, support for features such as mutual authentication and single sign-on, and resilience against established attack models. Our findings reveal that more recent protocols based on elliptic-curve cryptography generally offer improved security and performance compared to earlier approaches. Nevertheless, several protocols continue to exhibit weaknesses, including susceptibility to impersonation attacks, lack of perfect forward secrecy, and reliance on centralized authentication entities. Furthermore, our study also exposes significant inconsistencies in experimental setups and performance metrics across prior studies, which complicate fair and reproducible comparisons. Ultimately, our study concludes by identifying key gaps and research directions for the development of more robust and lightweight authentication mechanisms, as well as the establishment of standardized criteria for comparative evaluation in MEC environments.

**Keywords:** Authentication, Elliptic-Curve Cryptography, Multi-access Edge Computing, Network Security

## 1 Introduction

Cloud computing is able to satisfactorily provide processing, storage, and networking resources to devices at the network edge, thereby assisting resource-limited devices [1]. However, the cloud imposes important limitations, since its highly centralized architecture can lead to high latency and, consequently, to user

dissatisfaction. To tackle this issue, Multi-access Edge Computing (MEC) emerges as a promising alternative to the centralized cloud model by offering computational resources at the network edge to support latency-sensitive applications [1]. As a matter of fact, MEC enables faster application and service responses, improves user experience, and promotes more efficient utilization of network resources [2, 3].

Notwithstanding, network security still remains a major obstacle to the widespread adoption of MEC, as edge devices typically have limited computational resources and are unable to execute security algorithms as robust as those deployed in cloud data centers. Particularly, designing and evaluating protocols to ensure mutual authentication between users and MEC servers with strong security and resistance to attacks is a nontrivial task, given the characteristics of the MEC environment.

To further compound this challenging scenario, the existing academic literature on authentication protocols for MEC is fragmented, and many studies proposing new schemes evaluate them against existing solutions using non-standardized and potentially biased comparison methodologies. In this context, this article aims to combine a structured literature comparison with a neutral and systematic analysis of the main authentication protocols proposed for MEC, with the goal of guiding future research in this domain. As its main contribution, this article thus identifies open gaps in the literature, emphasizing the need for fairer and more consistent evaluation methodologies, as well as key design considerations that should inform the development of future MEC authentication protocols.

The remainder of this article is organized as follows. Section 2 reviews the main background concepts required to understand this work. Section 3 then presents and individually discusses each authentication protocol explored in this article, followed by Section 4, which compares them in terms of their strengths and limitations. Section 5 discusses open gaps and suggests directions for future research, and finally, Section 6 concludes the article.

## 2 Background

This section presents several concepts to help understand this study. More precisely, Subsection 2.1 introduces the fundamental idea of Multi-access Edge Computing. Subsection 2.2 explains cryptographic concepts essential for the remainder of this article, while Subsection 2.3 describes how authentication protocols operate. Finally, Subsection 2.4 introduces Game Theory and discusses how it can be applied to cybersecurity.

### 2.1 MEC

Multi-access Edge Computing (MEC) is an emerging paradigm proposed by the European Telecommunications Standards Institute (ETSI) that aims to complement cloud computing by moving data-processing servers closer to the network edge [4], creating an environment characterized by ultra-low latency, high communication bandwidth, and real-time access to local radio network information.

With the widespread use of mobile devices such as smartphones and laptops, data generated by these devices commonly travel hundreds of kilometers to be processed by centralized servers and then is sent back to the source device at the network edge. MEC envisages deploying data-processing servers throughout the network edge, for instance at radio access points that connect mobile devices. This approach allows data to be processed locally, significantly reducing latency and alleviating communication bottlenecks with centralized cloud servers. Moreover, placing servers at the network edge enables information sharing among nearby users, facilitating applications such as communication among autonomous vehicles within a neighborhood [5].

The physical distance of centralized cloud computing servers limits the responsiveness of real-time applications and services used by mobile devices. MEC increases the feasibility of real-time applications in mobile networks but relies on the deployment of geographically distributed infrastructure and still faces several challenges [4]. One major challenge is to ensure interoperability across multiple local networks, a process that is inherently dependent on the security of edge devices.

The network edge is a critical point where services are consumed, yet it is also one of the weakest points in terms of security. Many mobile and Internet of Things (IoT) devices are manufactured with resource-constrained hardware that cannot execute strong cryptographic and security computation. Edge devices are often vulnerable to cloning and physical tampering, which creates multiple attack vectors in MEC networks [4]. This significant security concern has been the focus of numerous studies in the literature, particularly research on authentication protocols, which are the main subject of the present article.

### 2.2 Cryptography

Cryptography is the science of transforming data, allowing information to be understood only by authorized parties. In modern contexts, it is closely associated with computing systems and Internet protocols, with

symmetric and asymmetric criptography widely used in the Internet. Nonetheless, cryptography is a very ancient discipline [6].

Asymmetric cryptography employs two keys, a public and a private one. One of them is used to encrypt a message and the other key, distinct but mathematically related, decrypts it. Asymmetric cryptography complements symmetric cryptography and is particularly useful for securely distributing symmetric session keys to all authorized parties in a communication [7].

Elliptic Curve Cryptography (ECC) is a modern asymmetric cryptographic technique. According to [8], ECC enables the use of smaller key sizes than those required by other asymmetric cryptosystems while maintaining an equivalent level of security. In addition to smaller keys, ECC typically requires less communication bandwidth and can be executed more efficiently by processors, making it especially attractive for applications with limited computational power and memory.

ECC relies on elliptic curves whose variables and coefficients are defined over elements of a finite field [7]. ECC operations, such as point addition and point multiplication, do not follow conventional arithmetic rules, despite sharing similar notation. An important operation used in ECC is the bilinear pairing, which is defined in [9] as a mapping function between two groups and a third group, that is, a function of the form $S \times T \rightarrow \beta$, where $S$ and $T$ are groups (for example, elliptic-curve groups of order $q$) and $\beta$ is a group of order $q^k$.

### 2.3 Authentication

Authentication is the process of ensuring that the identity of a communication partner is genuine [10]. An authentication protocol defines the sequence of message exchanges between two entities that enables such verification. In a typical authentication protocol, a user initiates communication with a trusted entity and, after some message exchanges, both parties become assured of each other's identities, even under the assumption that all exchanged messages could be observed, modified, or replayed by a malicious adversary.

The characteristics of devices under MEC significantly increase the challenge of secure authentication, since a robust authentication protocol in this scenario must satisfy the three fundamental pillars of security (confidentiality, integrity, and availability) despite the limited computational capabilities of MEC devices.

In addition to being secure and computationally lightweight, an authentication protocol should exhibit some desirable properties that enhance security or usability, such as perfect forward secrecy, known-key security, anonymous authentication, single sign-on (SSO), and independence from third parties during authentication.

Perfect forward secrecy is defined in [11] as the property whereby the compromise of cryptographic keys from a current session does not jeopardize the secrecy of keys from previous sessions. Differently, known-key security is the guarantee that compromising the cryptographic key of a current session does not allow an adversary to derive keys from future sessions [12].

Anonymous authentication is achieved when an adversary is unable to determine the identity of any party involved in a communication process based solely on the public parameters exchanged during authentication [12], and SSO allows a user to access and authenticate with multiple servers without repeatedly providing credentials such as usernames and passwords [13].

### 2.4 Game Theory

Game theory is a mathematical framework used to model decision-making processes involving two or more parties within a predefined scenario [14]. The theory was introduced by John von Neumann and Oskar Morgenstern in 1944 [15] with the original goal of modeling and predicting economic behavior. Although initially developed for economics, game theory was soon recognized as applicable to other fields, such as sociology, international relations, and political science [14], as well as network security.

Game theory studies the concept of a game, in which two or more entities, named players, capable of making decisions interact according to a set of predefined rules. The interactions among players lead to outcomes that depend on the strategies chosen by each participant [14]. Players may pursue different objectives: in some games, players cooperate to achieve a common goal, while in others they act adversarially to hinder their opponents.

When the players in a game have strictly opposing objectives, the gain of one player necessarily implies the loss of another. This class of games is known as zero-sum games [16] and constitutes the primary model adopted in the work reviewed in this article. The games discussed typically aim to model the interaction between an attacker and a victim or server, defining the rules and capabilities of each player. These games are zero-sum because the objectives of the participants are inherently conflicting: the attacker seeks to successfully compromise the system, while the victim aims to prevent any successful attack.

When applied correctly, game theory provides a powerful tool for security analysis, as it supports the evaluation of the robustness of authentication protocols by simulating realistic scenarios in which both

attackers and victims are intelligent agents capable of making strategic decisions.

# 3 Authentication Protocols

This section analyzes and discusses five representative authentication protocols for MEC. They were chosen for this comparison due to their dissemination in industry and academia. The key characteristics, advantages, limitations, and security weaknesses of each protocol are outlined. For ease of reference and organization, each protocols is denoted by the citation of its original publication and is discussed separately. In addition, they are all summarized in Table 1, which reports, for each protocol, the year of publication, the type of cryptography employed, the presence of single sign-on (SSO) support, and the hardware platform used for performance evaluation.

| Reference | Year | Cryptography | SSO | Hardware |
|---|---|---|---|---|
| [17] | 2015 | ECC | Yes | Intel Core2 Quad-core 2.40-GHz CPU<br>3GB RAM |
| [12] | 2016 | ECC | Not specified | Not available |
| [13] | 2019 | ECC | Yes | Server: Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30 GHz, 1 GB RAM and Ubuntu 14.04<br>Smartphone: Google Nexus One, 2 GHz ARM CPU armeabi-v7a, 300 MiB RAM, Android 4.4.2 |
| [18] | 2021 | ECC | Not specified | Intel Core I7-8550 U CPU 1.80 GHZ |
| [19] | 2019 | ECC | Yes | Server: Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30 GHz, 1 GB RAM, Ubuntu 14.04<br>Smartphone: Google Nexus One, 2 GHz ARM CPU armeabi-v7a, 300 MiB RAM, Android 4.4.2 |

Table 1: Authentication protocols.

**1) The protocol of [17]**

The protocol proposed in [17] is frequently referenced in the literature, with its limitations cited by several authors [20, 12]. The proposed authentication mechanism consists of two stages: the registration phase and the authentication phase. During registration, a trusted entity is required to generate smart cards for each user. These smart cards store user-specific identification information, and the Smart Card Generator (SCG) functions as a secure distributor of cryptographic keys.

Essentially, the SCG acts as a cryptographic key distributor. Therefore, although the authors considered scenarios involving smart cards, the proposed protocol is not restricted to this type of device. Any trusted entity capable of providing the same cryptographic keys may be used in place of the SCG, employing a technology other than smart cards.

The SCG, however, is not involved in the authentication phase. In this phase, the user sends a login request directly to a service provider, supplying the information stored on the smart card. Authentication involves only the user and the service provider, which makes the process faster by eliminating the need for additional entities. Because it does not rely on trusted third parties during authentication, this protocol is well suited for MEC environments: it avoids communication bottlenecks with external trusted entities and facilitates seamless connections for mobile users, while still preserving SSO functionality.

Furthermore, this protocol defines a sequence of message exchanges between the user and the service provider to achieve mutual authentication. This message flow, including all computations, is illustrated in Figure 1. In the first step of the authentication phase, the service provider uses a bilinear pairing function $e$ to compute $e(P, P)^a$, where $P$ is a prime number used in ECC and $a$ is a nonce. This message is then sent to the user, initiating the second step of the protocol. In this step, the user performs a series of computations to derive the session private key $K_{ij}$, as well as the messages $K_2$ and $C_1$, which make use of concatenation ($||$) and exclusive OR ($\oplus$, XOR) and will be used during the authentication. In the third step, the service provider also computes the session key $K_{ij}$, verifies the user's identity, and generates the message $D_i$. In the last step, the user computes $D_i'$ to verify the identity of the service provider.

The security of this protocol was assessed by using a game-theoretic approach. In the formal proof, the attacker's capabilities are defined, along with several possible attack vectors targeting different stages of the authentication process. The authors argue that, based on the defined capabilities, an attacker cannot successfully compromise the protocol and that authentication remains secure even under attempted attacks. However, despite this formal analysis, subsequent studies have identified security vulnerabilities in the protocol.

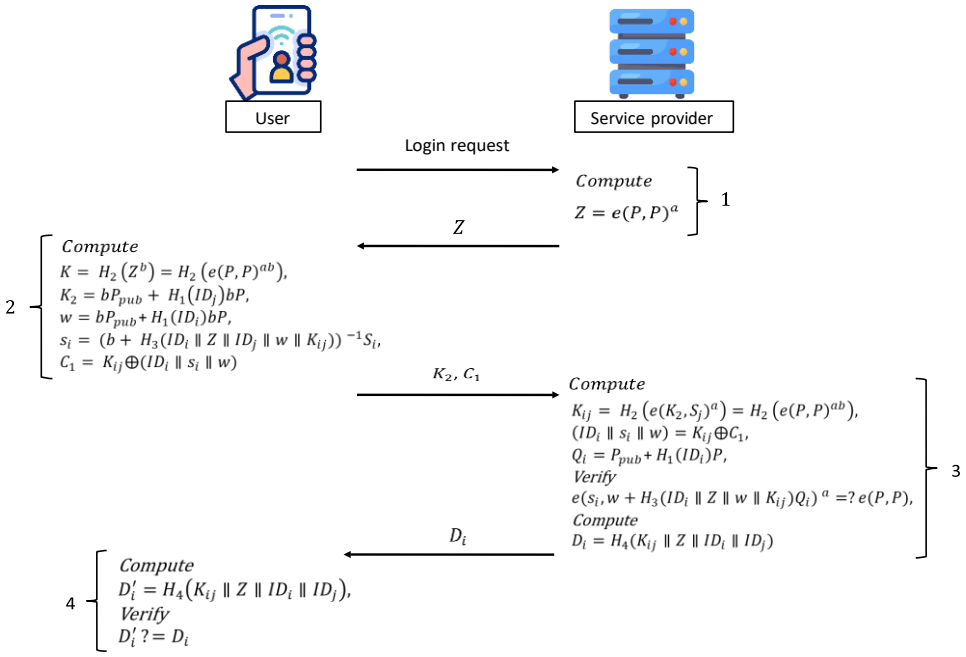



Figure 1: Message exchange to achieve mutual authentication in [17].

The work in [20] is an analysis of the protocol proposed in [17], where the authors argue that it is vulnerable to attacks of service provider impersonation and that it suffers from some design flaws. The impersonation attack takes place during the authentication phase and is illustrated in Figure 2. An adversary impersonated as the service provider forges a false message by altering part of the first message sent to the user. This modification enables the attacker to later compute the cryptographic session key, which grants access to messages subsequent to the authentication.

In addition to this severe security vulnerability, the authors of [20] identified three other, less critical, weaknesses in the protocol. The first concerns the improper use of biometrics: the protocol applies raw biometric data directly to a hash function, specifically in the second computation block (Figure 1), expression $K_{ij} = H_2(Z^b)$. Biometric acquisition does not always yield identical results, and hash functions are highly sensitive to minor variations in their input. As a consequence, a legitimate user may fail to authenticate due to small perturbations in the captured biometric data. The authors therefore suggest replacing the hash function with a fuzzy extractor, enabling authentication as long as the biometric samples are sufficiently close.

The remaining minor issues identified by [20] are the absence of incorrect password and biometric checks, which leaves the protocol susceptible to brute-force attacks, and the lack of a mechanism to revoke a lost or stolen smart card. Although qualitative solutions are proposed for these secondary issues, the authors of [20] do not provide a solution for the main impersonation vulnerability, leaving this as open work for future research.

### 2) The protocol of [12]

The authors of [12] also identified the weaknesses in the protocol proposed in [17], which, according to their analysis, may lead to impersonation and denial-of-service attacks. This time, however, the article proposes improvements intended to block these attack vectors by introducing a new authentication protocol. The message exchanges required to achieve mutual authentication in this protocol are shown in Figure 3. This protocol builds upon the one proposed in [17] and aims to compute the session private key $M_{ij}$ for both parties involved in the communication. Similar to its predecessor, the protocol includes registration and authentication phases, but it also introduces a password-update phase. The authentication phase is designed to protect the user against the impersonation vulnerability described earlier.

Block 1 (Figure 3) introduces the collection of the user's authentication information, including the identification number $ID_i$, password $PW_i$, and fingerprint $f_i$. These data are used to authenticate the user and to make the protocol resistant to dictionary attacks, in which multiple common passwords are tested, since the user's fingerprint adds substantial entropy to the authentication process. The value $D_i'$ is computed using the collected information, and the verification $D_i \stackrel{?}{=} D_i'$ ensures that only the legitimate user can access the system.

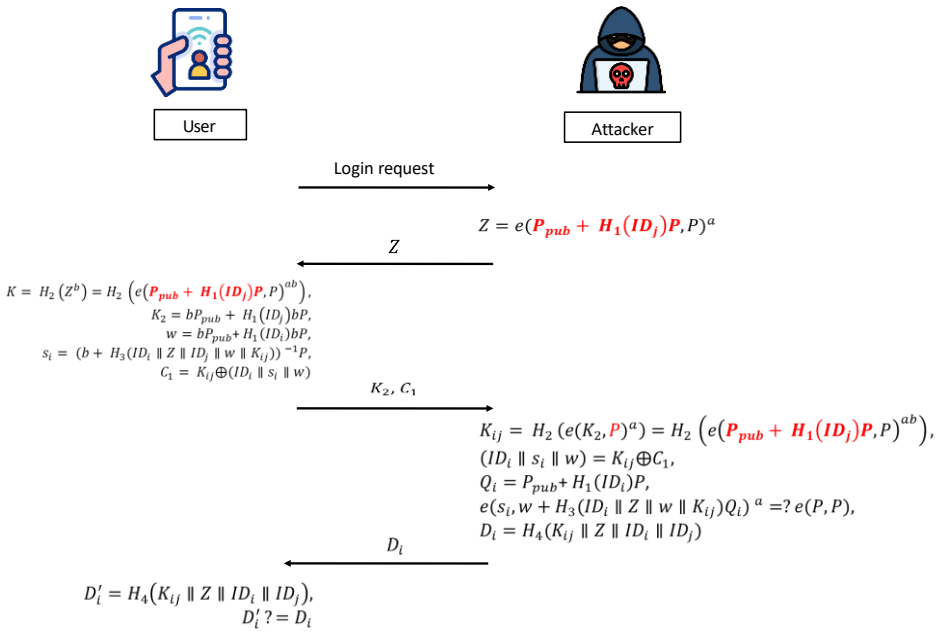



Figure 2: Impersonation attack against the protocol [17], as described in [20], with modified messages in red.

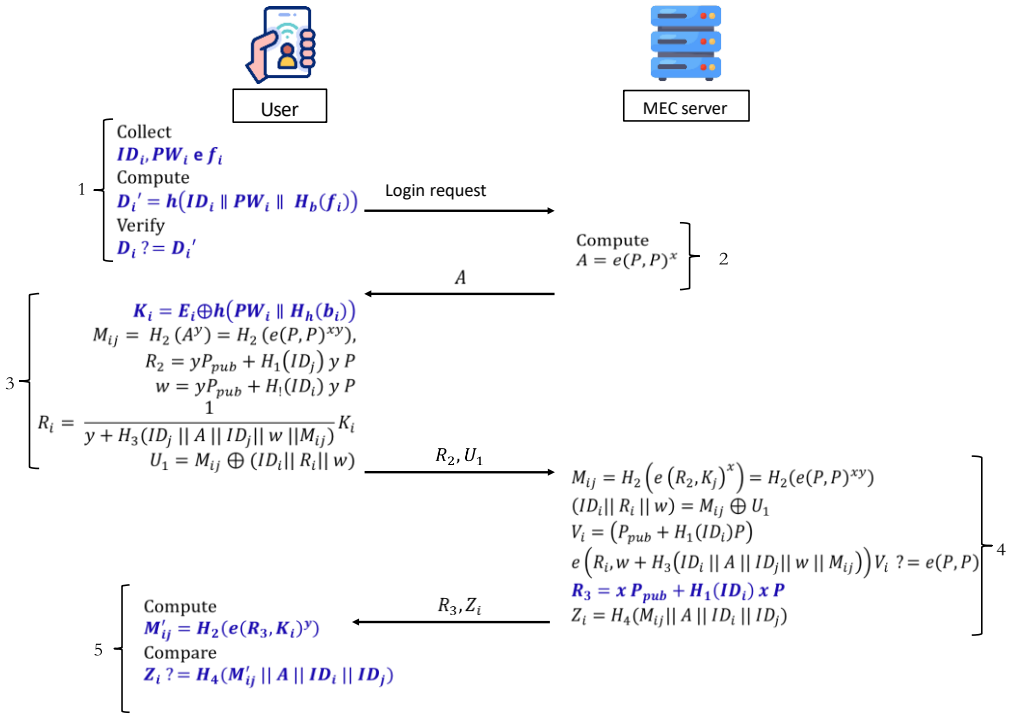
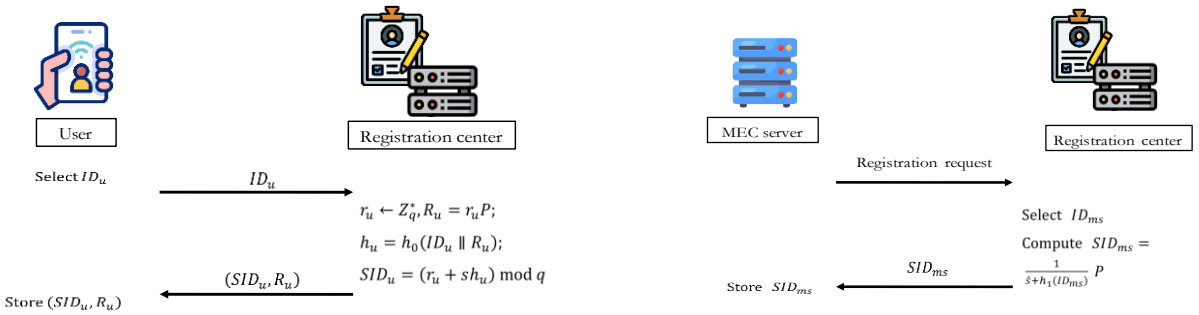



Figure 3: Message exchange to achieve mutual authentication in [12]. Text in blue corresponds to the elements added to the protocol originally proposed in [17].




(a) User registration. (b) MEC server registration.

Figure 4: Registration phase of the protocol in [13].

The equations highlighted in blue in Blocks 3, 4, and 5 of Figure 3 are used to prevent the impersonation attacks previously discussed. If an adversary attempts to send the message $A = e(P_{pub} + H_1(ID_j)P, P)^x$ in order to impersonate the legitimate server, the verification $Z_i \stackrel{?}{=} H_4(M'_{ij} \| A \| ID_i \| ID_j)$ will fail, and the authentication process will not succeed. The remainder of the protocol follow the same equations used in [17].

The security proof in [12] also relies on game theory, establishing the rules, capabilities, and objectives of the adversary. The protocol is then simulated in the presence of this adversary, and the authors considered its security to be formally demonstrated. However, the increased resistance to attacks comes at the cost of reduced computational efficiency, since additional mathematical operations are required to authenticate a user. Although this overhead mitigates security vulnerabilities, it is undesirable in MEC environments, where devices are subject to strict resource constraints.

### 3) The protocol of [13]

The authors of [13] proposed an authentication protocol that employs techniques similar to those used in the protocols discussed so far, such as elliptic-curve cryptography and bilinear pairings. This protocol preserves SSO functionality, and the authors claim that it is not vulnerable to impersonation attacks while also being more efficient than the protocol of [12]. This improvement is possible because the protocol in [13] is not a direct derivation of [17], as is the case with [12], but rather a novel design.

The protocol proposed in [13] also includes a registration phase and an authentication phase. The message exchanges during registration are shown in Figure 4. This protocol assumes the existence of a trusted third party, namely the registration center (RC). In this phase, the user must register with the RC by sending a unique identifier $ID_u$ and receiving a private key $(SID_u, R_u)$ in return (Figure 4a). The MEC service provider must also register with the RC, obtaining an identifier $SID_{ms}$ (Figure 4b). Although relying on trusted third parties is not ideal in MEC scenarios, this registration occurs only once and therefore should not introduce communication bottlenecks or delays in the authentication process.

Before registering any users, the RC must initialize the elliptic-curve cryptographic parameters. It selects a cyclic additive group $G$ over an elliptic curve $E/\mathrm{F}_p$ and a multiplicative group $G_T$, both of order $q$. The RC also chooses a bilinear map $e : G \times G \rightarrow G_T$, a generator point $P$, and computes $g = e(P, P)$. Next, it selects a private key consisting of two values, $s$ and $\hat{s}$, chosen uniformly at random from the group $Z_q^*$ and computes $P_{pub} = sP$ and $\hat{P}_{pub} = \hat{s}P$. Finally, six hash functions $(h_0, \ldots, h_5)$ are defined.

Although terminology varies among authors, [8] defines the group $\mathrm{F}_q^*$ as the set of values $\{1, 2, 3, \ldots, 1 - p\}$. With all parameters initialized, the RC publishes the tuple $(G, G_T, e, P, P_{pub}, \hat{P}_{pub}, g, h_0, \ldots, h_5)$ while keeping $(s, \hat{s})$ as its private master key.

Mutual authentication between the user and the MEC service provider in the protocol of [13] is accomplished in only three blocks, involving two message exchanges, as illustrated in Figure 5. The reduced number of message exchanges, compared with the protocol proposed by Tsai and Lo [17], can be advantageous in MEC environments because it decreases bandwidth usage and reduces the waiting time for responses from the service provider.

To initiate authentication, the user selects a random value $x \in Z_q^*$ and computes $g_x$, $X$, $M$, $N$, and $\sigma$ using the parameters published by the RC, the user's identifier $ID_u$, and the local timestamp $T_u$. The user then sends $(M, N, \sigma, T_u)$ to the MEC server. Upon receiving the message, the server checks whether $T_u$ is recent and rejects the request if it is not. After this verification, the server performs the computations required to check whether $\sigma P = W + h_5(ID'_u \| R'_u \| X' \| T_u) X'$. If this verification fails, the server must reject the login request.

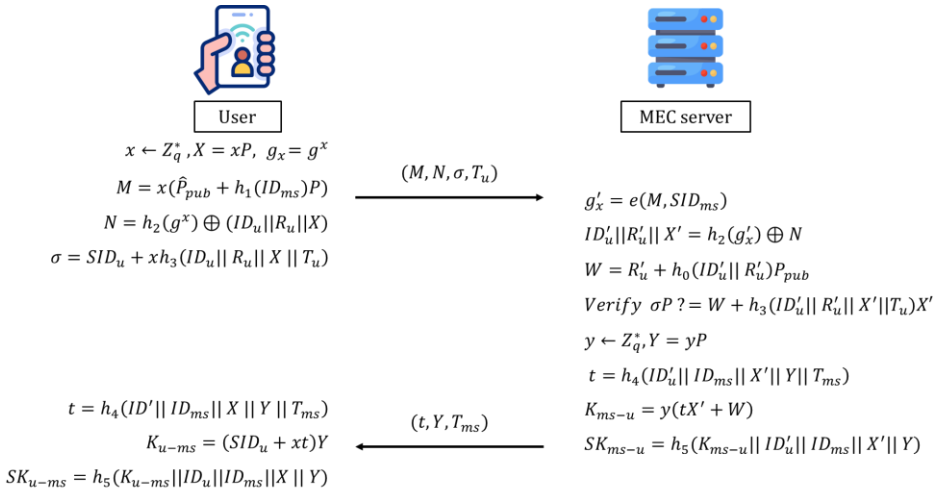



Figure 5: Message exchange in the authentication phase of the protocol in [13].

Upon verifying the user's identity, the server selects a value $y \in Z_q^*$, computes $Y$ and $t$, and sends the message ($t$, $Y$, $T_{ms}$) back to the user, where $T_{ms}$ is the MEC server's timestamp. The server also computes the session key $SK_{ms\text{-}u}$. The user then verifies the identity of the MEC server by checking whether $t = h_4(ID' \| ID_{ms} \| X \| Y \| T_{ms})$, and subsequently computes the session key $SK_{u\text{-}ms}$.

The authors of [13] simulated their protocol in order to compare its computational costs with those of the previously discussed protocols [17, 12]. In their evaluation, a cloud server was used to emulate the MEC server, while a smartphone was used to perform the computations on the user side. The simulation results will be presented in Section 4. Owing to the smaller number of message exchanges in the authentication phase and the lower computational cost of its operations, the protocol of [13] appears to be better suited for MEC environments than the earlier protocols. Nevertheless, this protocol has also received criticism from other authors.

Different studies [21, 22, 18] pointed out potential security weaknesses in the protocol of [13]. The authors of [21] stated that the main drawback of this protocol is its inability to manage the keys of MEC servers, while [22] claimed that the protocol is vulnerable to man-in-the-middle attacks, temporary parameter leakage, and does not provide perfect forward secrecy. However, neither [21] nor [22] presented evidence to validate these criticisms.

### 4) The protocol of [18]

Similarly, the authors of [18] mentioned several security issues in the protocol of [13], yet without offering supporting evidence. In view of that, they proposed a new authentication protocol for MEC. Their proposal aims to ensure the security of data transmitted over the network, preventing leakage and breaches of privacy, as well as the security of the application as a whole, minimizing risks at each communication link.

In this protocol, user authentication depends on communication with the RC, a design choice not found in the other protocols discussed here, which explicitly avoid involving the RC in the authentication phase to reduce communication latency and simplify the overall process. According to the authors, incorporating the RC into the authentication phase reduces the computational load on both the user and the MEC server. It also eliminates the need for the MEC server to store users' personal data, thereby enhancing the overall security of the architecture.

The mathematical foundations of the protocol follow those of [13], and the message exchanges in the authentication phase are shown in Figure 6. Using the hash function $h$, the identity extraction function *Gen*, and the identities *SID* and *ID*, it is possible to derive the session keys *SK*. The protocol is resistant to desynchronization attacks due to the timestamps $T_{ms}$ and $T_u$.

### 5) The protocol of [19]

Finally, another protocol that is frequently cited in the academic literature is the one proposed in [19]. Similar to the previously discussed protocols, it relies on elliptic-curve cryptography and hash functions to ensure secure authentication between the user and the MEC server. This protocol is specifically designed for MEC environments and aims at low computational cost and efficient session-key distribution, while not requiring an RC during the authentication process.

This protocol is divided into three phases: setup, user registration, and authentication. The authenti-

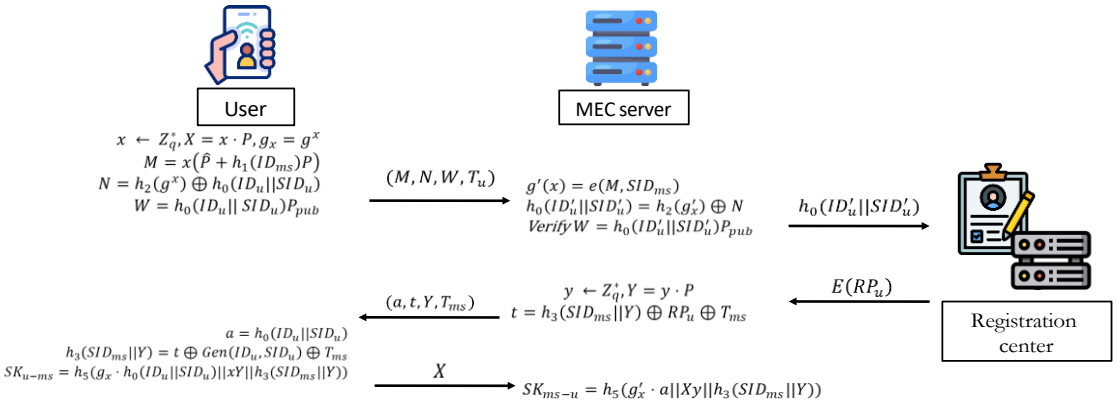



Figure 6: Message exchange in the authentication phase of the protocol in [18].

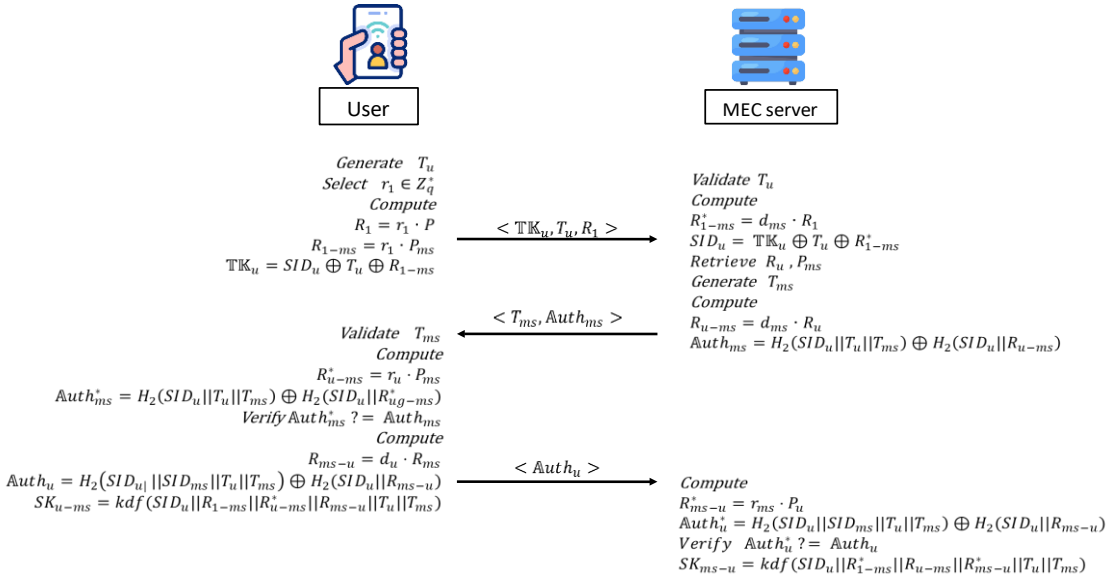



Figure 7: Message exchange in the authentication phase of the protocol in [19].

cation phase is the most critical, as it is executed repeatedly and is sensitive to delays. Its computations and message exchanges are illustrated in Figure 7. Desynchronization attacks are prevented through the use of the timestamps $T_u$ and $T_{ms}$. The required computations are performed to generate the authentication values $Auth^*_{ms}$ and $Auth^*_u$ which are then compared with the authentication messages sent by each party, thereby ensuring mutual authentication. The communication keys $SK_{u\text{-}ms}$ and $SK_{ms\text{-}u}$ are also computed to guarantee secure communication.

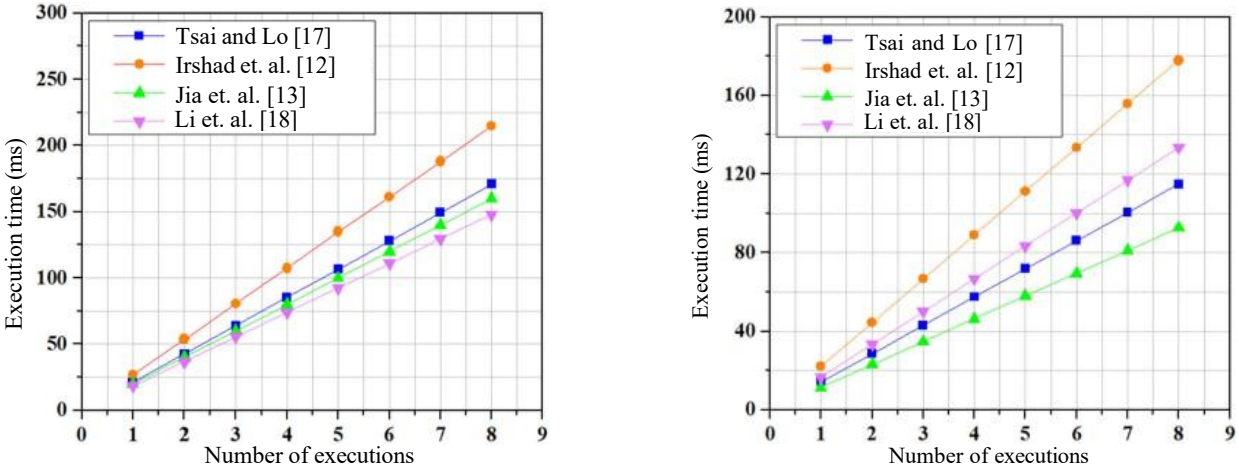
The security analysis conducted by the authors of [19] does not rely on game theory; instead, it explicitly describes which components of the protocol provide security properties such as mutual authentication, resistance to impersonation attacks, and resistance to man-in-the-middle attacks, among others. A game-theoretic analysis could enhance the credibility of the protocol's security claims, as in other papers. To date, this protocol has not received substantial criticism in the academic literature comparable to that directed at the protocol of [17], but only minor, unsubstantiated critiques, such as [23], which claims the existence of vulnerabilities if the MEC server is compromised by malware, without providing further explanation.

## 4 Comparative Analysis

This section compares the authentication protocols presented in the previous section, with particular emphasis on their similarities and limitations in light of results reported in prior studies. For clarity and conciseness, the comparison is structured around two complementary perspectives: a theoretical analysis (Subsection 4.1) and an empirical evaluation (Subsection 4.2).

| | [17] | [12] |
|---|---|---|
| Registration messages | $1 \cdot T_{PM}$ | $1 \cdot T_{PM}$ |
| User | $4 \cdot T_{PM}$ | $4 \cdot T_{PM} + 1 \cdot T_{BP}$ |
| Service provider | $2 \cdot T_{PM} + 3 \cdot T_{BP}$ | $4 \cdot T_{PM} + 3 \cdot T_{BP}$ |

Table 2: Time-complexity comparison between the protocols proposed by [17] and [12].




(a) User side. (b) MEC server side.

Figure 8: Comparison of cumulative processing time between authentication protocols performed by [18].

### 4.1 Theoretical comparison

As already mentioned, the work in [17] is the most widely cited and well-known reference and, consequently, also the most heavily criticized [20]. This work has served as a baseline for subsequent studies [12, 13, 18, 19], which aim to address its major vulnerabilities, particularly the service-provider impersonation attack.

The proposals by these authors focus on mitigating the vulnerabilities identified in [17] while also designing authentication protocols for MEC that are efficient from both algorithmic and practical perspectives. The authors of [12] presented a time-complexity analysis of their protocol in comparison with [17], with the results summarized in Table 2, where $T_{BP}$ denotes the time required to perform a bilinear pairing operation and $T_{PM}$ denotes the time required to perform a point-multiplication operation. It can be observed that, in order to address the shortcomings of [17], the protocol proposed in [12] incurs higher computational costs for both the user and the service provider.

An important issue observed in the reviewed protocols is whether they use an RC. Only the schemes proposed by [13] and [18] explicitly employ an RC. In the former, the RC is restricted to the initial registration and key-distribution phases, thereby avoiding additional latency during authentication. In contrast, the latter relies on the RC as an active participant in the authentication process itself, which, although reducing computational overhead on edge devices, may introduce communication delays that conflict with the low-latency requirements of MEC environments.

### 4.2 Empirical comparison

Beyond the theoretical comparison of protocols, some authors also provided simulation-based performance evaluations, such as [18]. Figure 8 presents the results of eight simulation runs for the protocols in [17, 13, 18]. On the user side (Figure 8a), the protocol of [18] achieves the lowest execution time. This is particularly relevant in MEC scenarios, where minimizing the computational cost on the user side can lead to lower battery consumption for smart devices that are not connected to a power grid, thus avoiding heavy processing on resource-constrained devices. On the MEC server side (Figure 8b), the protocol of [13] exhibits the lowest execution time.

However, the processing time incurred at the RC during the authentication phase is not reported, even though [18] is the only protocol that invokes the RC at this stage. Omitting the RC processing time may significantly bias the assessment of the protocol's real efficiency, potentially leading to misleading conclusions when selecting the most suitable protocol for a given MEC environment. Moreover, it is necessary to evaluate whether the time required for message exchanges between the MEC server and the RC is significant, or whether the primary bottleneck lies in the processing time at the user and MEC server.

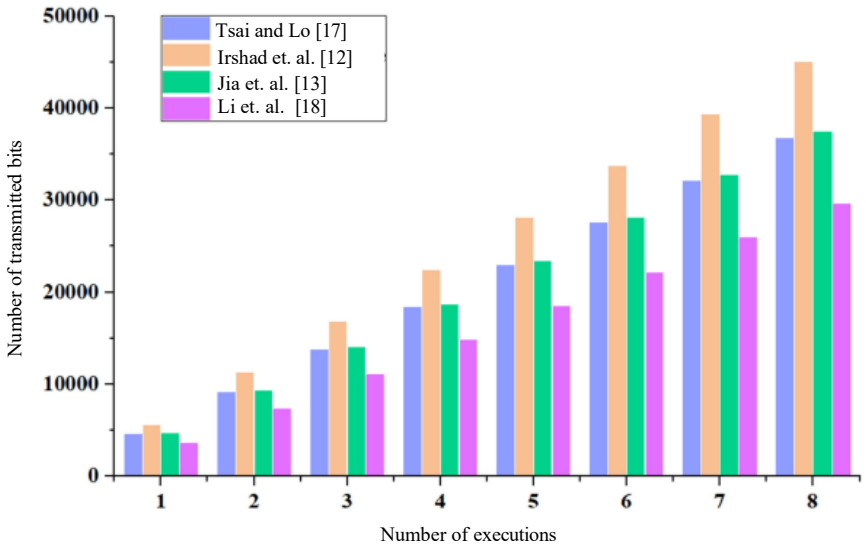
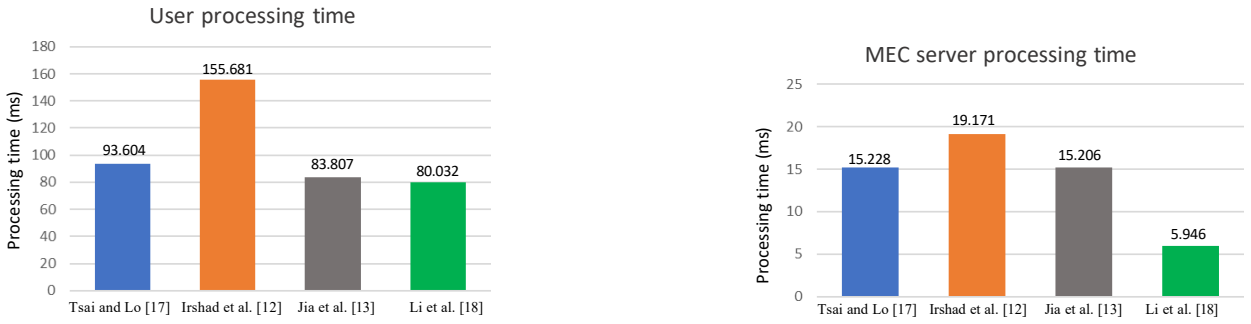



Figure 9: Cumulative bits transmitted during the authentication phase of the protocols compared in [18].




(a) User side (b) MEC server side

Figure 10: Theoretical execution time of the protocols compared in [19].

Communication overhead was also evaluated and compared in [18] by computing the cumulative number of bits transmitted during the authentication phase, with the results shown in Figure 9. Although the authors' protocol achieves the best performance under this metric, the analysis did not consider communication latency between the involved parties. One of the core principles of the MEC paradigm is to avoid communication with the cloud, which may be geographically distant from the user. If the RC is deployed on a cloud server, the communication latency between the MEC server and the RC may be significant, potentially resulting in authentication delays unsuitable for MEC environments.

Another comparative performance analysis is presented in [19], in which the authors computed the time required to execute several cryptographic operations used in the authentication protocol. Based on the measured costs of these operations, an algorithmic analysis is performed to estimate the total execution time of the protocol, as shown in Figure 10, with [19] yielding the lowest theoretical execution time. Nevertheless, a full simulation would be desirable, since the actual execution of an authentication protocol may take longer than the sum of the execution times of its individual components due to factors such as memory management, paging rates, concurrent processes, and other system-level effects.

## 5 Future Research Directions

In view of the analyses of MEC authentication protocols presented in this article, we outline, in this section, directions for future research in this area. Despite the contributions surveyed, selecting the most suitable authentication protocol for MEC remains challenging. While it is possible to identify specific strengths and limitations in each proposal, there is still no definitive conclusion regarding which protocol should be preferred. We argue that a more comprehensive and systematic comparison of these protocols is required to support such a decision.

A first comparison dimension concerns the security aspect, aiming to identify potential vulnerabilities in each design. However, this task is inherently challenging because the diversity of construction techniques

requires in-depth inspection of the underlying mathematical details, often demanding specialized expertise. In contrast, a performance-oriented comparison is more attainable. Such a comparison should implement all protocol components on the same hardware platform, enabling an objective assessment of efficiency in terms of computation and memory usage. Conducting this type of evaluation is nontrivial, as it requires not only a general-purpose programming language but also the implementation (or integration) of elliptic-curve cryptography primitives. During each simulation, key metrics should be collected, including execution time, number of message exchanges, energy consumption, and memory usage. With quantitative results, the resulting analysis would be clearer and more impartial.

Some widely used programming languages provide native support for ECC. For instance, Java and C# offer elliptic-curve classes available in both languages. However, these classes do not expose point-addition operations, which are heavily used in the protocols discussed in this work. C# also provides an ECC class tailored for Diffie–Hellman key exchange [24], which may be useful for implementing parts of these protocols. Another alternative is to rely on cryptographic APIs such as those provided by Bouncy Castle (https://www.bouncycastle.org), which offer a wide range of cryptographic algorithm implementations for both Java and C#. Python is another popular language; however, we were unable to identify native ECC support or libraries maintained by major organizations that provide comprehensive ECC functionality. Regardless of the chosen language or library, implementing cryptographic primitives from scratch is strongly discouraged due to their complexity and the high risk of introducing bugs.

Another promising direction for future work is the design of new authentication protocols for MEC. In this context, particular attention should be paid to the vulnerabilities highlighted throughout this article, such as impersonation attacks and the handling of biometric and personal data. Moreover, efficiency considerations are paramount in MEC environments. New protocols must account for the limited processing capabilities of user devices as well as the resource constraints of MEC servers. Finally, careful design of data exchanges is essential, whether between users and MEC servers or involving the RC. Beyond the volume of transmitted data, network latency, communication protocols, and the underlying physical transmission media should also be taken into consideration.

## 6 Conclusions

This article provided a critical comparison of authentication protocols for MEC, emphasizing not only the design but also the methodological soundness of their evaluation. The analysis shows that MEC authentication remains a challenging problem, requiring a trade-off between security properties, computational efficiency, and architectural constraints. Early work prioritized lightweight execution but exhibited serious vulnerabilities [17], whereas later proposals improved security at the cost of additional computation or architectural complexity [12].

A key contribution of this comparison is the identification of recurring methodological shortcomings in the literature, including non-standardized simulation setups, incomplete performance reporting, and limited consideration of communication latency, particularly when a centralized entity, such as the RC, is involved. These issues make fair comparisons difficult, and concealing the practical implications of each protocol.

Despite these challenges, the explored literature indicates a clear evolution of MEC authentication protocols, with work proposed in [13] standing out as a promising solution with a reduced number of message exchanges and absence of reported critical vulnerabilities to date. This article suggests future research combining rigorous security analysis with standardized and transparent performance evaluation, enabling a more reliable protocol decision for MEC environments.

## Acknowledgment

This research is part of the INCT of Intelligent Communications Networks and the Internet of Things (ICoNIoT) funded by the National Council for Scientific and Technological Development (CNPq), grant number 405940/2022-0, and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) grant number 88887.954253/2024-00. Icons from Freepik (www.freepik.com) were used in the figures.

## References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] R. A. Cardoso da Silva, C. K. da Silva Rodrigues, and V. Rocha, "A parameterized model for multimedia-content placement of the streaming service on 5g cellular networks," *Revista de Informática Teórica e Aplicada*, vol. 32, no. 2, p. 36–51, Mar. 2025. [Online]. Available: https://seer.ufrgs.br/index.php/rita/article/view/140750

[3] C. K. da Silva Rodrigues, V. Rocha, and R. A. C. da Silva, "Towards MEC-Servers Deployment for Multimedia Streaming over 5G CRAN-Architecture Networks," *IEEE Latin America Transactions*, vol. 23, no. 2, pp. 87–93, 2025.

[4] P. Ranaweera, A. D. Jurcut, and M. Liyanage, "Survey on multi-access edge computing security and privacy," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1078–1124, 2021.

[5] "Multi-access edge computing (mec)," Dispon´ıvel em: https://www.etsi.org/technologies/multi-access-edge-computing, acesso em: 19 mar. 2025.

[6] C. Paar and J. Pelzl, *Understanding cryptography*, 2010th ed. Berlin, Germany: Springer, Nov. 2014.

[7] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 6th ed. USA: Prentice Hall Press, 2013.

[8] A. H. Koblitz, N. Koblitz, and A. Menezes, "Elliptic curve cryptography: The serpentine course of a paradigm shift," *Journal of Number Theory*, vol. 131, no. 5, pp. 781–814, 2011, elliptic Curve Cryptography. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022314X09000481

[9] R. E. Blahut, *Cryptography based on bilinear pairings*. Cambridge University Press, 2014, p. 422–474.

[10] A. Tanenbaum and D. Wetherall, *Computer Networks*, ser. Pearson custom library. Pearson, 2013. [Online]. Available: https://books.google.com.br/books?id=w_d5ngEACAAJ

[11] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, Jun 1992. [Online]. Available: https://doi.org/10.1007/BF00124891

[12] e. a. Irshad, A., "An improved multi-server authentication scheme for distributed mobile cloud computing services," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 12, Dec. 2016.

[13] X. Jia, D. He, N. Kumar, and K.-K. R. Choo, "A provably secure and efficient identity-based anonymous authentication scheme for mobile edge computing," *IEEE Systems Journal*, vol. 14, no. 1, pp. 560–571, 2020.

[14] F. C. Zagare, *Game Theory and Security Studies*. Oxford University Press, 01 2019. [Online]. Available: https://doi.org/10.1093/oso/9780198831587.003.0002

[15] J. Von Neumann and O. Morgenstern, "Theory of games and economic behaviour," *Princeton University Press*, 1944.

[16] A. Attiah, M. Chatterjee, and C. C. Zou, "A game theoretic approach to model cyber attack and defense strategies," in *2018 IEEE International Conference on Communications (ICC)*, 2018.

[17] J.-L. Tsai and N.-W. Lo, "A privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Systems Journal*, vol. 9, no. 3, pp. 805–815, 2015.

[18] Y. Li, Q. Cheng, X. Liu, and X. Li, "A secure anonymous identity-based scheme in new authentication architecture for mobile edge computing," *IEEE Systems Journal*, vol. 15, no. 1, pp. 935–946, 2021.

[19] K. Kaur, S. Garg, G. Kaddoum, M. Guizani, and D. N. K. Jayakody, "A lightweight and privacy-preserving authentication protocol for mobile edge computing," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[20] Q. Jiang, J. Ma, and F. Wei, "On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Systems Journal*, vol. 12, no. 2, pp. 2039–2042, 2018.

[21] A. Bounceur, "A secure and lightweight zkp-based mutual authentication scheme with key agreement," *Arabian Journal for Science and Engineering*, 12 2024.

[22] O. Alruwaili, M. Tanveer, A. Saud, S. Alanazi, and A. Armghan, “Raaf-mec: Reliable and anonymous authentication framework for iot-enabled mobile edge computing environment,” *Internet of Things*, vol. 29, 12 2024.

[23] A. Bibi, M. Majeed, S. Ali, I. Abbasi, A. Samad, and S. Baseer, “Secured optimized resource allocation in mobile edge computing,” *Mobile Information Systems*, vol. 2022, pp. 1–14, 08 2022.

[24] Microsoft, “Ecdiffiehellmancng class,” https://learn.microsoft.com/en-us/dotnet/api/java.security.spec.ellipticcurve?view=net-android-35.0, [Ultimo acesso: 19 de janeiro de 2025].