electronics

MDPI

Article

Token-Based Authentication Framework for 5G MEC Mobile Networks

Wojciech Niewolski, Tomasz W. Nowak, Mariusz Sepczuk * and Zbigniew Kotulski

Institute of Telecommunications, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland; w.niewolski@tele.pw.edu.pl (W.N.); t.nowak@tele.pw.edu.pl (T.W.N.); z.kotulski@tele.pw.edu.pl (Z.K.)
* Correspondence: msepczuk@tele.pw.edu.pl

Abstract: MEC technology provides a distributed computing environment in 5G mobile networks for application and service hosting. It allows customers with different requirements and professional competencies to use the services offered by external suppliers. We consider a service access control framework on 5G MEC networks that is efficient, flexible, and user-friendly. Its central element is the MEC Enabler, which handles AAA requests for stakeholders accessing services hosted on the edge servers. The JSON Web Token (JWT) open standard is a suitable tool for the MEC Enabler to manage access control credentials and transfer them securely between parties. In this paper, in the context of access control, we propose the token reference pattern called JSON MEC Access Token (JMAT) and analyze the effectiveness of its available protection methods in compliance with the standard requirements of MEC-hosted services in 5G networks.

Keywords: 5G mobile networks; MEC; access control; 5G MEC security architecture; MEC Enabler; JWT

Citation: Niewolski, W.; Nowak, T.W.; Sepczuk, M.; Kotulski, Z. Token-Based Authentication Framework for 5G MEC Mobile Networks. *Electronics* **2021**, *10*, 1724. https://doi.org/10.3390/electronics10141724

Academic Editor: Taeshik Shon

Received: 25 May 2021
Accepted: 16 July 2021
Published: 18 July 2021

1. Introduction

The basic features that characterize 5G networks are the quality requirements for communication: high throughput, low latency, high device density, and high network reliability and availability, see [1]. These properties allow for the implementation of many network services previously unavailable in mobile networks. To facilitate the implementation of high-quality mobile network services dedicated to specific business domains and enable the provision of services by external providers, one integrated the MEC technology (Multi-Access Edge Computing) [2] with 5G. As a result, a new version of the mobile network has been created, namely 5G MEC [3,4]. It ensured the development of narrowly specialized web services and led to the concept of 5G MEC vertical industries, i.e., web service packages implemented in mobile networks and tailored to the requirements of different sectors of the economy [5].

In 5G MEC mobile networks, the interests of several independent groups of stakeholders meet. They use the facilities resulting from the quality parameters of the 5G network, the possibility of independent implementation of services offered by the MEC technology. They also take other advantages of solutions using 5G MEC networks, such as saving bandwidth and other network resources, resource latency, and increased security [5]. There are the following four main categories of service market participants in the 5G MEC mobile networks:

- **End-Users (EU)**. They are the parties who are using or will use the services provided in the 5G MEC environment in their professional activity or as consuming services.
- **Service Providers (SP)**. They are all entities that are deploying service solutions into MEC hosts. Most of them deliver MEC applications that might run in multiple instances, on multiple or even on a single MEC host server. The services hosted within the MEC Platforms can support multi-tenancy, can be isolated [6] or can cooperate and mutually share resources.

- **Network Providers (NP)**. They are physical (MNO—Mobile Network Operator) and virtual (MVNO—Mobile Virtual Network Operator) providers of wireless communications services that own or control the radio access layer and connect with external networks.
- **MEC Providers**. They are the entities that enable the MEC environment for 5G networks hosted by Network Providers. They can be both external organizations and Network Providers themselves.

The 5G MEC mobile networks can handle a wide range of use cases and services resulting from the needs of vertical industries [5]. Managing networks and services access control and supporting differentiation of availability becomes therefore necessary to maintain a stable state of the network during high traffic load with the necessary quality of service and constraints resulting from the security requirements of priority devices and services [7]. Therefore, the crucial security issue that must be addressed in 5G MEC networks is the management of security credentials, especially access permissions granted or obtained by the stakeholders assigned to different categories listed above.

There is much recent literature on authentication in mobile networks and access control systems. They show general access control schemes and detailed network-based authentication protocols. They draw attention to the presented solutions' functional and performance aspects as the number of steps, transparency, privacy-preserving, anonymity, suitability to mobile applications, or the Internet of Things (IoT). They also consider if a protocol provides mutual or multi-party authentication, is multi-factor, or includes authorization. However, in the context of our present studies, the crucial aspect is if the solution is suitable for MEC-hosted applications and 5G MEC-based services. In Table 1 we briefly present and compare several related papers which consider such solutions.

Table 1. Related authentication solutions. Applicability to 5G MEC: *—lowest, *****—highest.

Approach	MEC	Features	Summary
SDN-based authentication scheme [8]	****	secret key distribution, authentication	This paper proposes an SDN-based handover authentication scheme for MEC in cyber-physical systems. An authentication handover module in the SDN controller is applied for key distribution and authentication management. The parties apply the 3-way handshake protocol to achieve mutual authentication and secret key confidentiality.
Identity-Based Anonymous Authentication Scheme [9]	****	mutual authentication, anonymity, untraceability	The paper focuses on achieving mutual authentication with anonymity and untraceability. It presents a single message exchange round identity-based anonymous authenticated key agreement protocol for the MEC environment. The protocol achieves mutual authentication and assures both user anonymity and untraceability.
3rd-Party Authentication [10]	****	low latency, token-based	The paper proposes two solutions: Token-based Cookie transfer, 3rd-party Authentication and Token-based State transfer, 3rd-party Authentication for resolution of authentication and application mobility issues while achieving low latency.
Capability-based access control [11]	***	compatibility (OAuth 2.0, Verifiable Credentials)	The paper proposes a capability-based access control technique for sharing Web resources, based on Verifiable Credentials (VCs) and OAuth 2.0. It proposes a new form of OAuth 2.0 access token based on VCs.
AAA Security Design [12]	****	low latency, compatibility (OIDC)	The paper proposes a transparent security design called MECsec to secure the MEC with low latency in the cellular network. It contains three main components: cellular-based OpenID Connect (OIDC) authentication, bitmap-based authorization/accounting, and two-tier hash-based access control.
Privacy-Preserving Protocol [13]	***	lightweight, mutual authentication	This paper proposes a lightweight mutual authentication protocol for MEC environments based on Elliptic Curve Cryptography, one-way hash functions and concatenation operations.
Authentication Architecture [14]	***	lightweight, identity-based	The authors design a new authentication architecture for MEC environment, which provides an anonymous identity-based scheme suitable for lightweight devices.
Multi-Server Authentication Protocol [15]	****	mutual authentication, scalability	This paper proposes a user authentication protocol in the MEC environment providing anonymous mutual authentication between mobile devices and application servers operating on the MEC platform while ensures that the trusted MEC server can securely track users on behalf of service providers.
Scalable authentication and access control [16]	***	slicing support, IoT applicable	The paper proposes a Slice Specific Authentication and Access Control mechanism that makes use of the flexibility provided by virtualization technologies. This mechanism allows the authentication and access control of IoT devices to be delegated to the 3rd parties providing these devices.
5G reference security architecture [17]	***	general approach, risks identification	This paper gives the system reference architecture and overall security and privacy solutions for 5G applications. Based on the three major application scenarios of eMBB, uRLLC, and mMTC. It also provides specific suggestions for coping with security and privacy risks.
MEC Enabler-based solution [18]	*****	scalability, sovereignty, compatibility (OAuth 2.0), adaptability (ABAC)	The paper proposes the high-level security architecture of 5G MEC networks, which provides security solutions for the network's components and establishes secure access to all cooperating entities. The solution is based on the MEC Enabler, a new network's module, which manages security credentials required to access resources of MEC-hosted services.
This paper	*****	high security, compatibility (JWT, JWS)	For the security architecture proposed in [18], this paper defines the token reference pattern called JSON MEC Access Token (JMAT) suitable for MEC-hosted services in 5G networks.

A complex challenge in 5G MEC mobile networks is to provide a system for the distribution of access rights to resources and services that could meet extreme requirements expected by the stakeholders. Such a system should be efficient, scalable, and secure. In our earlier paper [18], we proposed the high-level security architecture of access control for 5G MEC mobile networks. We defined the components of the architecture and its security domains. Its crucial element is the MEC Enabler which implements access control management. We decided to use a security token system to transfer access credentials. Such a solution gives stateless, scalable, decoupled, transparent, and secure digital objects to transfer information among the network entities. Moreover, it is mobile-ready and suitable (in contradiction to cookies) for cross-domain applications. This paper concentrates on practical aspects of the proposed 5G MEC access control security architecture. First, we shortly present the model from paper [18], supplemented with details of information transfer between network modules in the authentication protocol. Next, we propose a complete JWT-based (JSON Web Token) system of access control management dedicated to 5G mobile networks using MEC technology. Finally, we present an empirical evaluation of components we use to provide secure access to 5G MEC systems. We present our results in the following steps:

- We give an outline of the 5G MEC security architecture and its security domains.
- We specify the authentication protocol which uses the MEC Enabler to generate and distribute security tokens.
- We present how the proposed "MEC Enabler" fits in the 5G architecture and its functions in the 5G MEC security architecture.
- We describe several existing JSON tokens schemes and the JMAT, which is the proposed token version suitable for 5G MEC access control service.
- Finally, we empirically compare the size of JMAT tokens and the time taken to generate and encrypt these tokens with an assortment of encryption algorithms and signing modes. It checks if the application in the proposed solution does not disturb MEC services' quality requirements, such as expected latency or data transfer overhead in 5G networks.

The rest of the paper is organized as follows. Section 2 presents the outline of 5G MEC security architecture, introduces the security areas in the 5G MEC mobile network and its central element for access control, the MEC Enabler. Section 3 presents an overview of security credentials suitable for use in the MEC Enabler for the authorization process of different stakeholders in the 5G MEC architecture. It also includes the state of the art and analysis of JWT-based security solutions as examples of general token-based applications for secure network authentication. Section 4 refers to the specification and characteristics of the JSON MEC Access Token (JMAT) with the division of its structure components and possible methods of its protection. The MEC Enabler uses the described token for representing security claims between a user and a service. Section 5 gives results of performance tests made on different hardware platforms to estimate the computational efficiency of the proposed JWT-based security solution. The prepared test shows computation time dependence for selected protection algorithms with representative parameters. The last Section 6 outlines conditions to practically apply the MEC Enabler with JMAT as an access control tool in 5G MEC. In this Section, there are also directions for future development of the security architecture for access control in the 5G MEC mobile networks and potential improvement for JMAT implementation.

2. The Outline of the 5G MEC Access Control Architecture and the MEC Enabler

Proposed in this paper token-based authentication framework is an element of the complete security architecture of 5G MEC considered in paper [18]. This section will briefly present the main components of this framework, focusing on the domain responsible for access control to MEC server-hosted services. Our security architecture model consists of ten security domains that coexist and cooperate, providing security services in all aspects of the 5G MEC mobile network functioning. Let us remark that such an approach is an

extension of the access control-oriented 5G security architecture defined in the paper [19], which now covers, additionally to 5G networks, the MEC technology requirements.

Thus, the 5G MEC security architecture consists of the following ten security domains, see [18]. Below we present their definitions and short descriptions, see Figure 1.

- **NAS** (**Network Access Security**): this domain guarantee security of user data by protection (confidentiality and integrity) of signaling and user data. It is realized for Control Plane (CP), Data Plane (DP), and User Equipment (UE) network.
- **NDS** (**Network Domain Security**) this domain is responsible for the protection of signaling and user data exchange between several network areas.
- **UES** (**User Equipment Security**): it is responsible at the user's side for hardware and software security, including user access to the mobile equipment.
- **AS** (**Applications Security**) it is responsible for the protection of communication (secure) between applications in UE and applications offered by the Service Provider. It is under the control of the end-user or the Service Provider (in the Application sub-Plane).
- **AKM** (**Initial Authentication and Key Management**): the security features that enable network functions to communicate securely. It implements authentication and key management mechanisms, which allows the realization of the unified authentication framework.
- **SCM** (**Security Credentials Management**) includes the relevant key management and service authentication between UE and the external data-transmitting network. In our model, the MEC Enabler governs this security domain.
- **SIO** (**Security Interoperability**) (also through MEC) this domain supports the openness of security capability between external Service Provider and the 5G network entity. It also includes a set of features that allows the stakeholders to know the status of security features.
- **NSS** (**Network Slices Security**) it is responsible for security of slices, for example for isolation, authorization and access control.
- **MECS** (**MEC Security**): it is responsible for the protection of software that belongs to the Service Provider, the hardware used for the realization of the MEC host and edge server, and for virtualization platform (VM).
- **CLS** (**Cloud Security**) includes all solutions to protect resources and communication inside the home domain (both the operator's and the cloud). This domain extends the resources offered by MEC-hosted services to the external cloud resources.

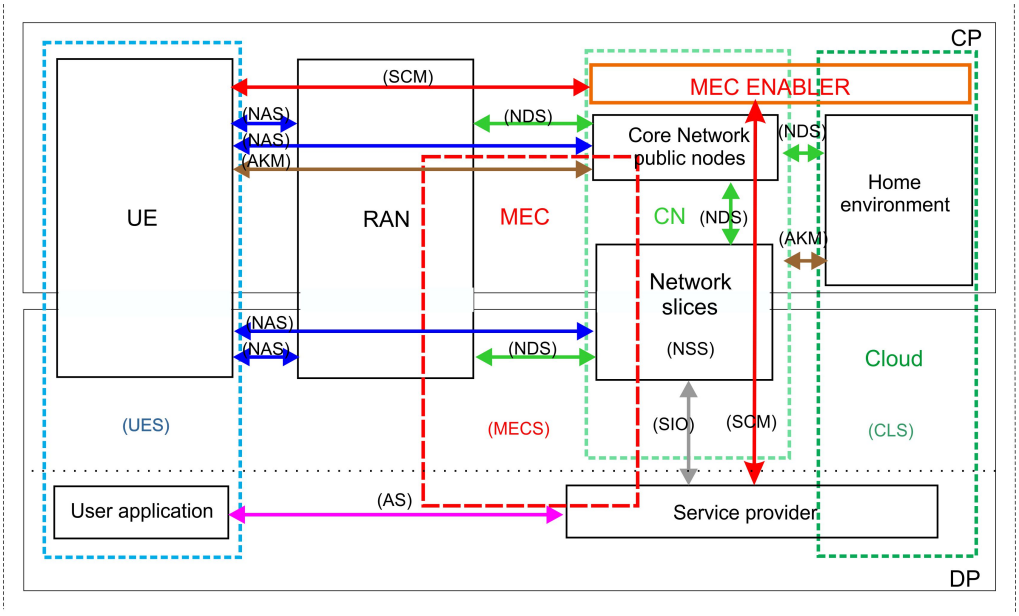

Figure 1. The high-level access control security architecture in 5G MEC.

The number of the domains we consider is higher than those in [20], where six security domains for the 5G network are defined. The security architecture proposed in [18] integrates four network environments: 5G CN (Core Network), 5G RAN (Radio Access Network), MEC (Multi-access Edge Computing), and a new module—the MEC Enabler. The MEC Enabler places the role of an AAA server in front of the MEC (it verifies requests). The MEC Enabler is also responsible for creating access tokens, which are necessary for the configuration of the MEC network. After positive authorization and token creation, it is possible to establish a connection with MEC infrastructure and requested services. The whole architecture is presented in Figure 2. The mentioned procedure, which is based on access tokens, is also one of the solutions which are analyzed and accepted by ETSI (European Telecommunications Standards Institute) [20]. One of the recommended forms of token implementation is JSON (JavaScript Object Notation) Web Tokens with digital signature protection, see the series of the RFC documents: [21–25]. Integration of the MEC Enabler in the MEC environment will reduce the management procedure and decrease the usage of all resources for non-allowed connections at the beginning. The MEC Enabler will be responsible for service authentication and credentials management for the UE and network's components, realizing:

- services for users which are properly authorized in the network,
- services for the protected communication (via protected links or slice),
- main security service which allows access to the MEC applications and services,
- management of access rights (for example by tokens or credentials),
- giving exchange credentials process for services realizing 5G MEC use cases,
- enabling services over the cloud or external operator's domain.

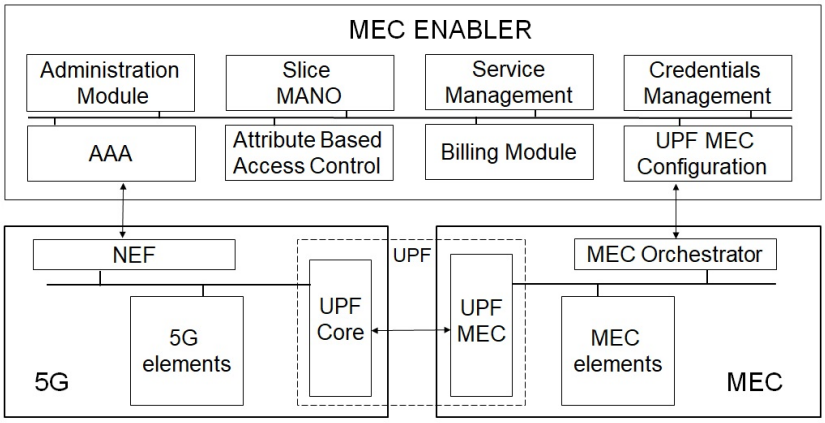

Figure 2. The new network architecture with the MEC Enabler.

The MEC Enabler extends some of the roles from the MEC; for example, it stores access policy based on some additional context such as information about clients, network slices, and others. Moreover, implementing a token-based authorization method by the MEC Enabler allows it to track all requests to the MEC resources and become a trusted party for several networks and MEC operators. On the other hand, mentioned functionalities needs from the MEC Enabler its integration with Network Exposure Function and MEC Orchestrator. Communication between MEC Enabler and MEC Orchestrator is necessary to verify MEC applications policies and privileges and control internal network access in the MEC area (UPF MEC). Thanks to this interface which can be realized by Common API Framework (CAPIF) [26] it is possible to establish a connection with MEC applications and configure all network elements. Communication between the MEC Enabler and the NEF is needed mainly for UPF Core configuration; its specification is presented in the 3GPP (3rd Generation Partnership Project), but it can also be done by CAPIF interface. Connection with both sides (edge computing and network) allows the MEC Enabler to store some necessary information about the MEC use and slice management. Therefore,

the MEC Enabler can send management commands for both slice and MEC services if necessary. Because mentioned functionalities are likely independent, for its realization, the MEC Enable needs to contain several functional blocks which cooperate to complete the access control service for the MEC-hosted applications. These modules are, see Figure 2:

- **Administration Module**: this module supports all management operations, and its responsible for configuration of other modules.
- **Slice MANO** (Management And Network Orchestration): this module is responsible for the management of the slice creation, updates, binding, and other operations which are connected with a slice life cycle. Another functionality of it is the reservation of slice resources for services, according to network segment information, service type, or any others, [27].
- **Attribute-Based Access Control**: the primary role of this module is to hold and expose policy information that can be related to MEC services, network access, slices, and others defined policy information from the MEC ecosystem [28].
- **Service Management**: this module is responsible for verification of available services, their current use, and the number of resources dedicated to them. Using this module will also make it possible to manage the MEC's services lifecycle using the MEC orchestrator API in some specific use cases. However, management functionalities will be available after implementation of an additional load balancer, and analyzer of service placement [29].
- **Credentials Management**: it creates tokens used for authorization of the MEC resources. It is also responsible for token refreshment or deactivation according to the MEC service needs [30].
- **Billing Module**: the role of this module is to keep billing information and collect information about MEC usage. Information from this module can be used for the realization of different business models (for example, resource reservation or usage) [31].
- **UPF MEC Configuration**: this module is responsible for UPF MEC configuration, including proper network slices binding [32]. According to the network policy, it can also match operators slices with MEC resources in some use cases.
- **AAA**: the role of this module is authentication, authorization, and accounting of all requests which are sent to MEC services [33]. If verification done by this module is positive, other modules start preparing network configuration by generating access tokens and API communication. Otherwise, if verification is negative, network configuration will not be applied, and finally, access to the MEC resources will not be possible. It is one of the main functionalities, which significantly improves the protection of network and MEC resources by the MEC Enabler.

The MEC Enabler can contain more logical blocks. However, this paper focuses on realizing AAA service with Credentials Management and their role in the 5G MEC secure ecosystem. By default, we assume that the AAA module receives the correct information about the network sliced structure, the services offered, and the appropriate security policy with its security parameters. We focus our presentation on the syntax of security credentials, their protection against security attacks, and the Credentials Management processing efficiency. A more detailed explanation of the MEC Enabler role in the 5G authentication process is presented in Figure 3, and [34] and all these steps are included in Algorithm 1:

Algorithm 1 The authentication procedure with the MEC Enabler, see Figure 3.

PROCEDURE BEGIN:

1. Registration request is sent from User Equipment (**UE**) to Access and Mobility Management Function (**AMF**).
2. Start of the Primary Authentication between **UE** and Authentication Server Function (**AUSF**) according to the 5G procedure [35].
3. **UE** establishes an NAS security context with **AMF**.
4. **UE** initiates the establishment of a new PDU Session by sending an NAS message.
5. **AMF** selects **V-SMF** (Visited Session Management Function) and sends the PDU Session context request.
6. **V-SMF** sends a PDU Session context response.
7. **H-SMF** (Home Session Management Function) obtains subscription information from the UDM and verifies that **UE** request is compliant.
8. **H-SMF** sends an EAP Request/Identity message to **UE**.
9. **UE** sends an EAP Response/Identity message.
10. **H-SMF** selects **UPF** and establishes N4 Session.
11. **H-SMF** forwards request to **UPF**.
12. **UPF** forwards the request containing EAP Response/Identity message to the MEC Enabler AAA server (**ME:AAA**).
13. **ME:AAA**, after verification, sends a token request to the MEC Enabler Credentials Management (**ME:CREDENTIALS MANAGEMENT**).
14. **ME:CREDENTIALS MANAGEMENT** sends tokens to **ME:AAA** needed for further authorization to the MEC.
15. **ME:AAA** authorizes a new configuration for the UPF Core (via the Network Exposure Function), allowing establishing a data plane connection between **UE** and **UPF CORE**.
16. **ME:AAA** sends information about an authorized session to the MEC Enabler UPF MEC CONFIGURATION module (**ME:UPF MEC CONFIGURATION**).
17. **ME:UPF MEC CONFIGURATION** module configures **UPF MEC** (via the MEC Orchestrator) to establish a new data plane connection to the MEC application (**MEC:App**).
18. **UE** establishes an End-to-End connection with the requested **MEC:App**.

PROCEDURE END:

To summarize, the MEC Enabler is a high-level concept applicable in different ways. Microservices and modular monolith could be used as a primary design pattern for building such a service. Good scalability should be provided to manage current and future traffic and the number of devices and incoming requests per second. CQRS (Command Query Responsibility Segregation) with loose data consistency and caching patterns should apply to archive the MEC Enabler's proper performance in general. This paper does not consider exact implementation details for the MEC Enabler, especially related to deployment. It only introduces the main MEC Enabler modules and the functions they perform to define the basics of the implementation requirements of the respective modules.

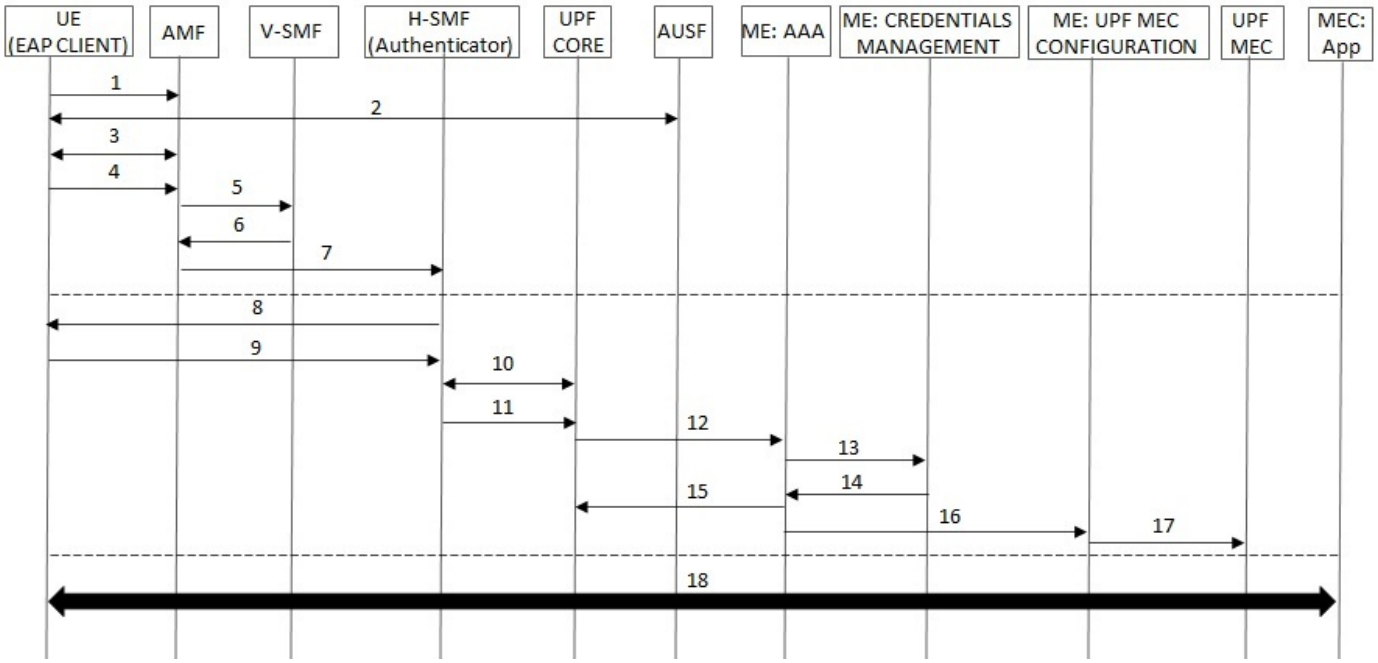

Figure 3. The authentication procedure with the MEC Enabler (Algorithm 1).

3. The Overview of Security Credentials Suitable for MEC Enabler

One of the core functionalities of MEC Enabler is an authorization of operations requested by different entities to multiple edge services. Moreover, this process also includes configuration of network access after positive verification [18]. Authorization for a large combination of parameters can be done using the Attribute-Based Access Control (ABAC) model. It provides grained access control for large-scale systems [36] and satisfies the most important expected requirements of MEC computing architecture [37]. Therefore, ABAC is widely used in many distributed systems such as Cloud Computing [38,39], Big Data [40,41] or Internet of Things [42–44] and authors of MEC Enabler also decided to use it. For proper operation of the ABAC system, it is needed to have a definition of information structure which will be exchanged between entities and verified with main policy [45]. One of the famous and proven implementations of secure access token which is used for different applications and platforms [46] is JSON Web Token (JWT) [25].

According to the specification [25], "JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties". The JWT uses JavaScript Object Notation (JSON) to represent these claims, and the result is a small and straightforward token used by protocols such as OpenID Connect 1.0 or OAuth 2.0. Generally, the JWT can be used to:

- Propagate identity between interested parties.
- Propagate user permissions between interested parties.
- Transfer data in a secure way between interested parties over an unsecured channel.

The reasonable questions are when we should use JWT and why? The first case that should be considered is authorization. It is the most common scenario for using JWT. Once the user is authenticated, each request will include the JWT, allowing the user to access routes, services, and resources permitted with a particular token. Single Sign-On is a feature that widely uses JWT presently because of its small overhead and ability to be easily used across different domains. Solution based on JWT can have only one authorization server that deals with Login/Registration and generates the token. In this approach, all the subsequent requests will not have to go to the authorization server as only the Auth-server will have the private key, and the rest of the servers will have the public key to verify the signature. It is useful for corporate systems where the authorization server is in a secure environment, e.g., a user needs to be connected to the intranet to log in. However, once

they are logged in, the public servers can verify and proceed on. A similar setup can be used for OAuth implementation. The best part is that there is no connection between the auth-server and the rest of the servers other than the pre-defined public key. The second usage case of JWT is Information Exchange. JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed, for example, using public/private key pairs, it assures the entity's authenticity and the non-repudiation of its actions. Moreover, because the signature is calculated using the header and the payload, verifying if the content has been tampered with is also possible.

The JWT (see Figure 4) defines the token format and uses additional specifications to handle signing and encryption. This collection of specifications is known as JOSE (JavaScript Object Signing & Encryption) and three of the most popular components are JSON Web Token (JWT) [25], JSON Web Signature (JWS) [21], and JSON Web Encryption (JWE) [22]. JWT is a standard format of token that can contain the signed or encrypted payload. When a token is signed, it applies JWS; when encrypted, it applies JWE. Token format of JWTs includes parts separated by a dot character (“.”). Each part is additionally encoded in Base64. The first part, called JOSE Header, specifies the type of JWT (JWS or JWE) and the used cryptographic primitives.

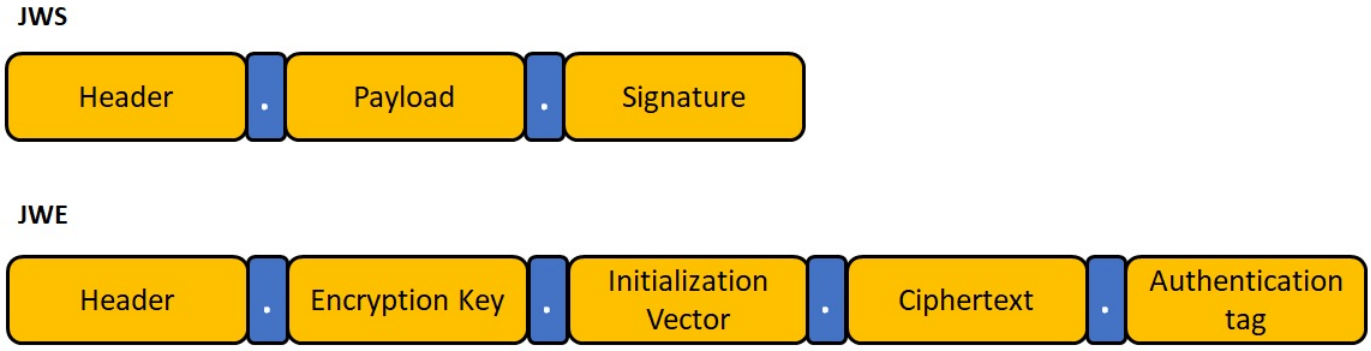

Figure 4. JWS and JWE components.

The JWS token consist of three parts: the JOSE Header, the JWS Payload, and the JWS Signature. The JWS Payload includes defined information, called claims, such as a username, user IDs, user roles, and many more. Finally, the JWS Signature is a hash of header, payload, and secret. JWS claims are signed with a signature (private key) that the interested parties can verify with a secret signing key (public key). It ensures that the claims have not been tampered with when transferred between parties. The contents of the JWS token are Base64 encoded and not encrypted. Thus, the JWS should be used to exchange non-sensitive information in the claim or the token payload. Thanks to the JWS approach, the rogue client or Man-in-the-middle can see the data as its Base64 encoded. If the rogue client or Man-in-the-middle tries to change data, the signature verification will fail. The server will validate the message's signature to ensure that the client did not modify the claims. If the server detects any modification, it can take appropriate action, such as deny the request or block the client.

The JWE scheme encrypts the content (JWT claims) and creates JWE Authentication Tag for the claims with the authenticated encryption algorithms. Thanks to the JWE approach, the rogue client or Man-in-the-middle cannot see the data because of encryption with a secret key. If the rogue client or Man-in-the-middle tamper the payload, the JWE Authentication Tag verification will fail, and the server will discard the signature. Man-in-the-middle attacks cannot modify it since the verification would fail. Naturally, the JWE [22] and JWA [24] standards offer many cryptographic algorithms that can be used. In this work, we focused on the ones selected to show which will best meet the requirements for token generation by MEC Enabler.

Usage of JWT allows defining a very complex policy with a dynamic number of entities and parameters. On the other hand, according to the authors of [47,48] and our results of experimentations described in the following sections, an increasing number of attributes significantly impacts on time for token creation and verification.

One of the methods to decrease the time of token verification was proposed in [49], and it assumed to use an alternative policy searching operation called ABAC-SRM. Another option for minimizing the time of token processing can be using a shorter key or a different protection method. Results describing the relation of token time validation for different key sizes and protection methods can be found in [50] and for the specific python-based implementation in Section 5 of this paper.

Subject of alternative or additional encryption methods for ABAC tokens was mentioned in [51] where the authors proposed usage of blockchain to improve their protection. Another proposed security extension to the ABAC model is C-ABAC [52] which provides additional encryption for objects and policy in distributed IT platforms. Moreover, implementation of JWT and ABAC technologies is presented in cloud or application services and starts appearing in network configuration systems. The needs of dynamic changes in Software-Defined Networks (SDN) and complex and growing policy can be resolved using the mentioned access control methods [53]. It can be suitable both for access control at the edge of the mobile network [54], and at network nodes along the flow path [55].

Presented examples of JWT token usage for many platforms and systems and the methods and securing them, confirm its maturity and rightness of using this technology in the MEC Enabler solution. Moreover, authors of [56] propose an architecture that connects multi-tenancy cloud solutions and applies a similar idea of protection. MEC is very close to cloud solutions, which is why our proposal to implement the MEC Enabler with JWT and ABAC system is an opportunity to manage authorization at the Edge Computing and be in line with current trends, too.

Therefore, the token structure has been adequately adjusted to the requirements of MEC services. The JMAT has been developed as a JWT dedicated to usage in MEC. Its detailed structure will be discussed later in this article.

4. The Design of JWT Authentication Token Used by MEC Enabler (JMAT)

The central part of the JSON MEC Authentication Token is a payload. Listing 1 shows an example token, which consists of the following fields:

- **id**: an identifier of the entity that sent the request (principal).
- **actor**: type and context of the principal, possible values are defined by each application independently, e.g., user, tester.
- **token_preferences.life_time**: the timespan while the token can be accepted. It might be shorter than the **(time_of_end_of_life - time_of_generation)** value to support the token refreshment.
- **token_preferences.nonce**: the random value that prevents the reply attack.
- **appid**: a unique name of the application.
- **slice.id**: an identifier of the slice used by the service inside the application.
- **slice.slice_type**: an array of attributes describing the slice that might be used by the application or by the control plane.
- **slice.service**: an array of services' identifiers that can be accessed by the principal using this slice. Those services are part of the application that is mentioned in this JMAT token.
- **access**: authorization access decision, for the scope of this article, it is the Boolean value, but might be extended to another set of possible decisions.
- **time_of_generation**: timestamp of the response generation, might be saved as an integer number.
- **time_of_end_of_life**: the access is valid before this timestamp, might be saved as an integer number.

Listing 1: The structure of the payload of JMAT.

```
{
"id":"John",
"actor":"user",
"token_preferences":{ "life_time":"123456","nonce":"123456789"},
"appid":"Application1",
"slice":[{
"id":"Slice1",
"slice_type":["eco","secure"],
"service":["Service1","Service2", "Service3","Service4"]
},{
"id":"Slice2",
"slice_type":[{}],
"service":["Service5", "Service6", "Service7"]
},{
"id":"Slice3",
"slice_type":[{}],
"service":["Service8","Service9", "Service10","Service11"]
}],
"access":"Granted!",
"time_of_generation":1606339425998,
"time_of_end_of_life":1606339426898
}
```

The payload of the token is created based on the user's policy file. This file contains data about the user and the application to which the user has access. The user rights to the application are closely related to the structure of the MEC application (see Figure 5).

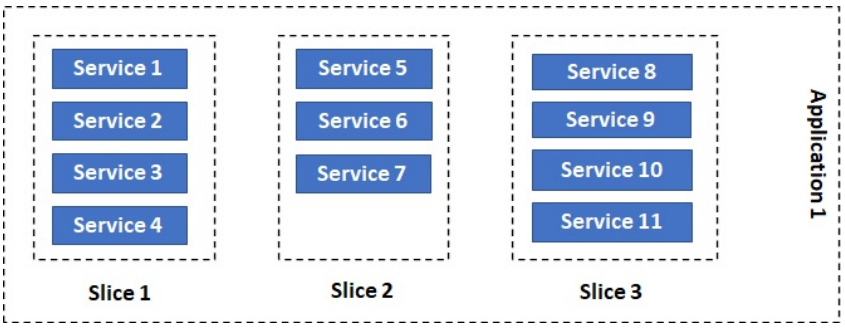

Figure 5. The MEC Application structure.

Every application has a unique identifier representing the type of access to it (restricted or anonymous) and slices deployed among the application. It can include many simple services responsible for a particular functionality (e.g., billing operations, management, etc.). There is also a possibility to store parameters for selected services in the user's entry. Those parameters are encoded with Base64 format and might contain, e.g., API keys to the service. Moreover, such a structure allows other tokens to be nested in a basic token and used in access control between services located in different slices. The policy file is delivered to the MEC Enabler from the MEC Environment Orchestrator. The article does not focus on creating this file in the MEC environment (for the research, it was assumed that the finally generated file is delivered to MEC Enabler).

Naturally, created tokens must have properly protected content. In our simulations we consider five versions of tokens using various encryption and token signing modes listed in RFC:

- Unprotected (marked as **U** in the Tables): the payload of JMAT represented as plaintext—only for input data size compare,

- encrypted with AES-CBC, signed with HMAC using SHA-256, AES Key Wrap [24] (marked as **A128**),
- encrypted with AES-CBC and signed with HMAC using SHA-256, symmetric key encrypted with RSA-OAEP [24] (**RSAC**),
- signcrypted with AES-GCM, symmetric key encrypted with RSA-OAEP [24] (marked as **RSAG**),
- only signed with HMAC using SHA-256 [21] (marked as **SHA**).

We selected such modes because we would like to check the efficiency of high-speed and short-lifetime solutions (SHA), a typical symmetric-based solution recommended by [24]—A128. Additionally, we would like to compare results with RSA-based solutions as an example of an asymmetric cryptography approach. All of them are supported by python library *jwcrypto* [57]. It uses *cryptography* library [58] and as a default cryptography backend (cryptographic primitives provider and implementation) the OpenSSL library [59] that supports AES-NI instructions [60].

In Table 2 there are sizes of tokens containing access credentials for a different number of applications and slices and with a different number of parameters.

Table 2. The size (in kB) of JMAT for different number of applications, slices, and parameters: before and after signcrypting or signing only.

No.	No. of Apps	No. of Slices	No. of Par.	U	A128 [kB]	RSAC [kB]	RSAG [kB]	SHA [kB]
1	16	64	64	335.84	246.77	247.05	247.03	447.87
2	32	64	64	335.84	246.67	246.95	246.93	447.87
3	64	16	64	84.12	62.27	62.55	62.53	112.24
4	64	32	64	168.05	123.75	124.03	124.01	224.15
5	64	64	16	87.48	63.04	63.32	63.31	116.72
6	64	64	32	170.25	124.23	124.51	124.49	227.08
7	64	64	64	335.87	246.87	247.16	247.14	447.90
8	64	64	128	667.01	490.44	490.72	490.70	889.42
9	64	64	256	1329.25	977.14	977.43	977.40	1772.41
10	64	128	64	671.42	492.25	492.53	492.52	895.30
11	64	256	64	1342.62	983.77	984.05	984.03	1790.24
12	128	64	64	335.81	246.48	246.76	246.73	447.83
13	256	64	64	335.85	246.64	246.93	246.91	447.88

We see that these numbers strongly affect the size of a token. For specific applications, one must consider an optimal choice of these numbers to optimize an overhead in access control data transmission. Another conclusion from the data given in Table 2 is that complete protection reduces the data size since proposed encryption and signature standards apply data compression with the Deflate algorithm. In contrast, data signature alone does not use such an additional algorithm. The signcryption strongly reduces data transmission overhead; however, it increases a computational overhead (see Table 3).

Table 3. Time of JMAT generation and verification (64 applications, 64 slices, 64 parameters) calculated for 500 samples, i9 10900 processor.

Alg.	key gen. [ms]	std. dev. [ms]	var.	gen. [ms]	std. dev. [ms]	var.	valid. [ms]	std. dev. [ms]	var.
A128	0.02	0.004	0.20	11.33	0.083	0.01	4.65	0.371	0.08
RSAC	62.76	42.671	0.68	11.43	0.448	0.04	5.49	0.374	0.07
RSAG	56.73	39.116	0.69	11.03	0.249	0.02	5.12	1.144	0.22
SHA	0.02	0.006	0.27	2.22	0.814	0.37	5.84	1.052	0.18

The choice of the algorithm may depend on the type of service the user wants to access. For example, anonymous access does not require a high level of protection. Nevertheless,

the generation of an appropriate token format and the use of a cryptographic algorithm requires time, which in the context of a large volume of traffic from users may be a bottleneck. Thus, performance testing is essential in such a situation.

5. Generation and Validation of JMAT: Numerical Results

Performance tests have been done within Virtual Machine with Linux OS. Tests were implemented in the Python language (version 3.9). For time measurements the function `time.perf_counter_ns()` was used. The resolution of the counter in such a configuration is 100 ns.

Table 3 presents basic statistical parameters (a mean value m, a standard deviation σ, and a variation ratio $\nu = \sigma/m$, see [61]), for three performance parameters related to JMAT, which are time of key generation, token generation, and token validation. The key generation process uses pseudo-random number generators, so the variation between the time generation of keys is quite high. It is evident for RSA methods, where it is necessary to test the primality of the generated numbers. As in the 1st and 4th row of Table 3, the symmetric key generation time is small compared to the token generation time. This is not the case with asymmetric cryptography (rows 2 and 3 in Table 3). For the most frequently used key length of 2048 bits, the token generation's key generation ratio is around 5. Table 4 shows the time of RSA key generation for several key lengths and the corresponding token generation time. The first time grows with key length (exponentially with the number of keys to choose from), known from the RSA theory. We see that for very short keys (substandard length of 1024 bits), the time of key generation time is comparable with the time of token generation. Thus, the key and the token can be generated together. For the key length of 3072 bits, which is recommended for "top secret" by the American Committee on National Security Systems (see [62]), the key generation process dominates over the token generation. Thus, we have the following recommendations for the key generation for JMAT.

- For symmetric cryptography, key generation can be done together with token generation.
- For asymmetric cryptography, key generation needs its separate computational thread or even a separate key generation server.
- For security reasons, the key generator in the MEC Enabler platform can be implemented without special hardware or software solutions, but it must be especially protected against data leakage.

The time of tokens generation and tokens validation for tokens of a fixed structure (number of components) in most cases is stable. The variability parameter is below 0.05 for generation and around 0.15 for validation. A more significant variability parameter for SHA generation comes from a tiny result to measure. External effects such as thread switching have a substantial impact than in more time-consuming scenarios. The token generation time linearly depends on the token's length and used algorithm. The token validation time slightly depends on the token's protection method.

Table 4. Time of asymmetric key and JMAT generation (64 applications, 64 slices, 64 parameters).

Key Size [Bits]	Key Generation [ms]	Token Generation [ms]	Ratio
1024	12.33	11.08	1.11
2048	59.75	11.23	5.32
3072	226.81	11.34	20.00
4096	545.61	11.35	48.05
8192	6670.35	11.68	571.33

Figure 6 shows the total time spent on the key generation, token generation, and token validation. Analogously, Figure 7 shows the time spent on the key generation and token generation. The algorithm A128 has better performance than RSAC and RSAG. The SHA algorithm without encryption is even faster than the A128 algorithm. However, the SHA algorithm does not provide confidentiality of the message. RSA-based algorithms are up to

5-times slower than A128. The token generation algorithm spends $O(1)$ time generating the token, where n is the number of applications available for a particular user. It was archived using index-based data storage.

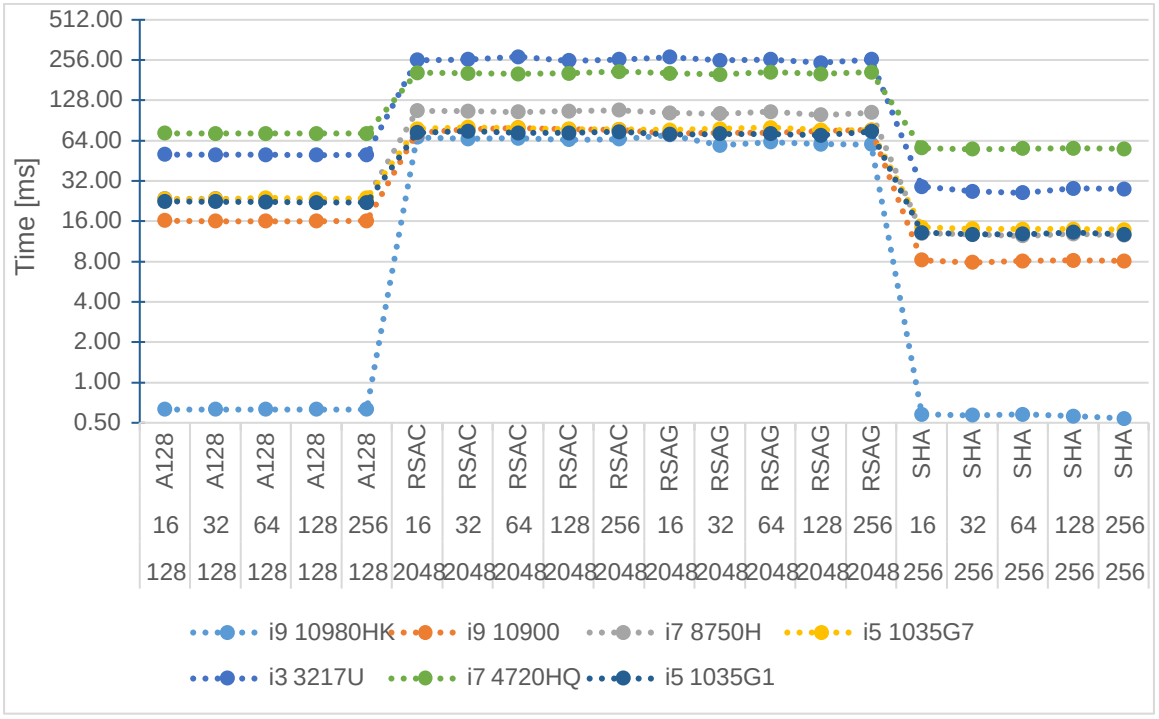

Figure 6. The total time for tested algorithms, for slices = 64, parameters = 64, and different number of applications. On the horizontal axis from the top respectively: algorithm, number of applications available for user during JMAT generation, and key length in bits.

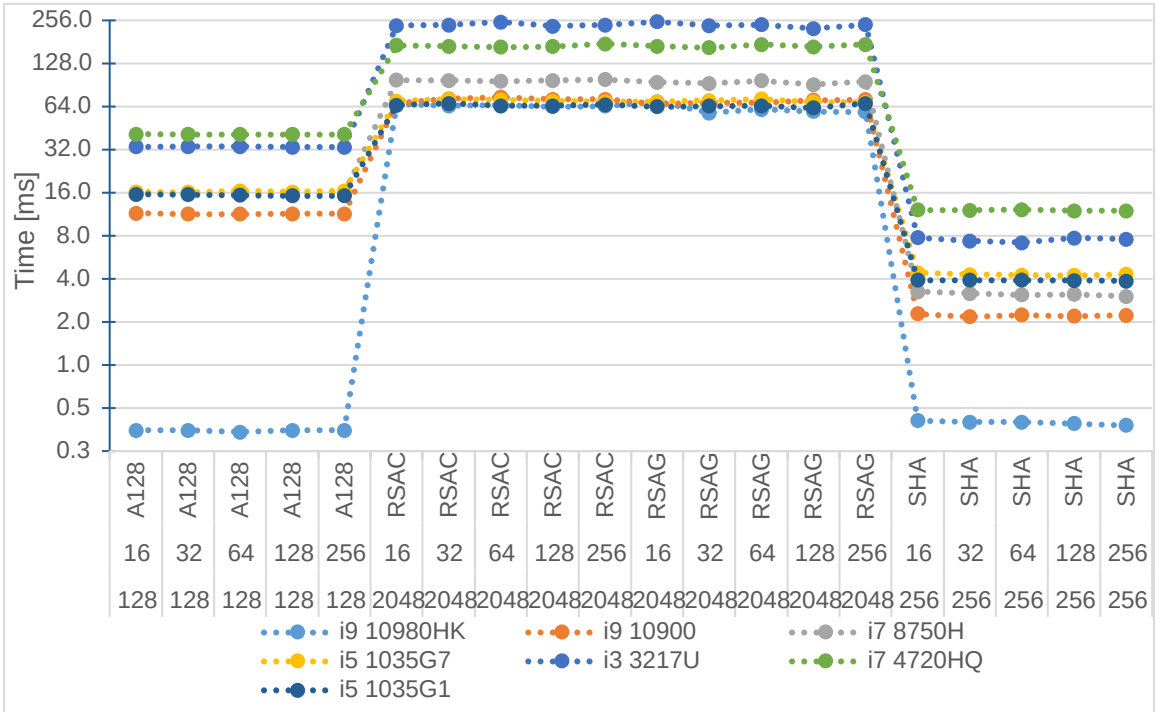

Figure 7. The key and token generation time for tested algorithms, for slices = 64, parameters = 64, and different number of applications. On the horizontal axis from the top respectively: algorithm, number of applications available for user during JMAT generation, and key length in bits.

The size of a created token is a valid metric for multiple use cases because even simple operations such as storing or transferring the token depend on its size. Table 2 one can read that output tokens can have smaller and larger sizes than the original payload. Using the compression turned on for all encrypted JMAT scenarios can provide a smaller token than the original payload. The used library [57] does not support the compression for the SHA algorithm, so the plaintext was uncompressed in this scenario. The Deflate algorithm was used because the standard supports it. Using compression provides obvious transport and storage benefits; it increases security as well. The compressed plaintext has higher entropy per bit than the non-compressed version of the message. When entropy per bit rises, then the plaintext is less predictable for an attacker. From this perspective, there is no substantial security argument for compressing the plaintext with the SHA scenario because it does not encrypt the plaintext.

The token has some overhead from the Base64 encoding, which allows the transfer of the token within text protocols such as HTTP. This encoding, unfortunately, adds about 33% to the total token size. Small tokens have a relationally significant overhead that comes from the JWE and JWS standards.

To summarize the process of token-based authentication in 5G MEC architecture, there is a need to mention one particular parameter, which is very important for this edge environment-latency. According to assumptions for 5G and defined specific use cases for MEC [63], this parameter should be reduced even to a value of 1 ms. Of course, this limitation is mainly for a response from service after establishing the connection, and service operation time is not included in this value. On the other hand, an authentication process will impact the whole process and balance security level and time consumption. Analysis of our work shows that using JMAT, it is possible to achieve a good time for token validation, even with a single CPU thread and python instructions. When the token validation process is done with a multi-thread program executed on a standard CPU equipped with 4 cores (8 threads), this time can be close to 0.6 ms. Another aspect for reducing the time of token creation and validation, which should be considered in the future, is a programming language used for those operations. Therefore, our experimentations prove the possibility of using the token-based method for authentication in MEC Enabler, even though there are still areas of possible improvement.

6. Conclusions and Future Work

Realizing services based on the 5G MEC paradigm will impressively improve data transmission quality and network efficiency compared to a typical 5G mobile network. On the other hand, usage of MEC technology will also bring challenges, the need for more resource protection and control, and especially requirements for higher security levels.

Due to many stakeholders in the edge service provision model and at the same time the dynamic ability to launch services, the edge computing system should allow for authentication based on multiple information: owner, service id, type of request, and many others. The other aspect of edge computing is resource limitations, so it is also essential to reduce the number of service requests to only those adequately authenticated. One way to limit unwanted connections to the MEC infrastructure and at the same time increase security is to use the MEC Enabler. This element establishes a connection to the MEC infrastructure only after successful authentication of the request. Both the MEC and the network resources are used only for valid requests.

Furthermore, such a function of access control based on different levels (network/application) requires an authentication mechanism based on many parameters. One possible realization for a complex AAA system is the ABAC method. As part of this document, state of the art was done for various systems using ABAC and JWT for network authentication. In addition, it was proposed to include it in MEC Enabler and as a form of private transmission to use a token compatible with the JWT model. Moreover, the authors of this document defined necessary fields for the MEC Enabler and named it JMAT. Analyzes of JMAT token for authorization of the MEC resources were done and proved.

Additionally, to check the possible security methods for JMAT and analyze their use in the edge model, performance tests were done. It was assumed to validate various combinations of the key lengths and encryption methods. Performance tests have been launched on different platforms. They have shown that even using python-based software, the speed of generation and verification of a multi-attribute token is sufficient for use in the MEC Enabler in a case when those operations will be done on a multi-thread program. In further works, it is also planned to compare the results using the latest libraries written in other programming languages, allowing multiple threads simultaneously. Interesting might also be checking results based on Elliptic Curve Cryptography, which might have different performance results for the key generation. In addition to extending the performance tests, the authors of MEC Enabler plan to develop its additional functionalities and attempt to integrate them with the MEC ecosystem.

Author Contributions: General 5G MEC architecture, Z.K.; MEC Enabler structure, W.N.; JMAT structure, M.S.; software, W.N., M.S. and T.W.N.; authentication protocol specification, W.N.; coordination of calculations and software optimization, T.W.N.; related work studies, formal analysis, visualization, writing—original draft preparation, writing—review and editing, W.N., T.W.N., M.S. and Z.K.; project administration, Z.K. All authors have read and agreed to the published version of the manuscript.

Funding: This paper has been supported by The National Center for Research and Development, Poland, under Decision No. DWM/POLTAJ7/9/2020.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

3GPP	3rd Generation Partnership Project
5G	fifth generation cellular network technology
AAA	Authentication, Authorization and Accounting
ABAC	Attribute-Based Access Control
AES	Advanced Encryption Standard
AKM	Initial Authentication and Key Management
AMF	Access and Mobility Management Function
API	Application Programming Interface
AS	Applications Security
AUSF	Authentication between UE and Authentication Server Function
CAPIF	Common API Framework
CDC	Cipher Block Chaining
CLS	Cloud Security
CN	Core Network
CP	Control Plane
CPU	Community Protection Units
CQRS	Command Query Responsibility Segregation
DP	Data Plane
EAP	Extensible Authentication Protocol
eMBB	enhanced Mobile Broadband
ETSI	European Telecommunications Standards Institute
EU	End-Users
HMAC	Hash Message Authentication Code
H-SMF	Home Session Management Function
JMAT	JSON MEC Access Token
JOSE	JavaScript Object Signing and Encryption
JSON	JavaScript Object Notation
JWE	JSON Web Encryption
JWS	JSON Web Signature
JWT	JSON Web Token

MANO	Management And Network Orchestration
MEC	Multi-Access Edge Computing
MECS	MEC Security
mMTC	massive Machine Type Communication
MNO	Mobile Network Operator
MVNO	Mobile Virtual Network Operator
NAS	Network Access Security
NDS	Network Domain Security
NP	Network Providers
NSS	Network Slices Security
OAEP	Optimal Asymmetric Encryption Padding
PDU	Protocol Data Unit
RAN	Radio Access Network
RSA	Rivest Shamir Adleman algorithm
SCM	Security Credentials Management
SDN	Software-Defined Network
SHA	Secure Hash Algorithm
SIO	Security Interoperability
SMF	Session Management Function
SP	Service Provider
SRM	Security-Related Metadata
UDM	Unified Data Management
UE	User Equipment
UES	User Equipment Security
UPF	User Plane Function
URL	Uniform Resource Locator
uRLLC	ultra-Reliable Low-Latency Communication
VM	Virtualization Machine Platform
V-SMF	Visited Session Management Function

References

1. *Minimum Requirements Related to Technical Performance for IMT-2020 Radio Interface(s)*; Report ITU-R M.2410-0; ITU: Geneva, Switzerland, 2017.
2. *Multi-Access Edge Computing (MEC). Framework and Reference Architecture*; ETSI GS MEC 003 V2.1.1; ETSI: Sophia Antipolis, France, 2020. Available online: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.02.01_60/gs_MEC003v020201p.pdf (accessed on 23 May 2021).
3. Blanco, B.; Fajardo, J.O.; Giannoulakis, I.; Kafetzakis, E.; Peng, S.; Pérez-Romero, J.; Trajkovska, I.; Khodashenas, P.S.; Goratti, L.; Paolino, M.; et al. Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN. *Comput. Stand. Interfaces* **2017**, *54*, 216–228. [CrossRef]
4. Hu, Y.C.; Patel, M.; Sabella, D.; Sprecher, N.; Young, V. Mobile Edge Computing. A key technology towards 5G. *ETSI White Pap.* **2015**, *11*, 1–16.
5. Nowak, T.W.; Sepczuk, M.; Kotulski, Z.; Niewolski, W.; Artych, R.; Bocianiak, K.; Osko, T.; Wary, J.-P. Verticals in 5G MEC-Use Cases and Security Challenges. *IEEE Access* **2021**, *9*, 87251–87298. [CrossRef]
6. Kotulski, Z.; Nowak, T.W.; Sepczuk, M.; Tunia, M.A. 5G networks: Types of isolation and their parameters in RAN and CN slices. *Comput. Netw.* **2020**, *171*, 107135. [CrossRef]
7. Li, J.; Della Penda, D.; Sahlin, H.; Schliwa-Bertling, P.; Folke, M.; Stattin, M. An Overview of 5G System Accessibility Differentiation and Control. *arXiv* **2020**, arXiv:2012.05520.
8. Wang, C.; Zhang, Y.; Chen, X.; Liang, K.; Wang, Z. SDN-Based Handover Authentication Scheme for MEC in Cyber-Physical Systems. *IEEE Internet Things J.* **2019**, *6*, 8692–8701. [CrossRef]
9. Jia, X.; He, D.; Kumar, N.; Choo, K.-K.R. A Provably Secure and Efficient Identity-Based Anonymous Authentication Scheme for MEC. *IEEE Syst. J.* **2020**, *14*, 560–571. [CrossRef]
10. Ali, A.; Lin, Y.-D.; Li, C.-Y.; Lai, Y.-C. Transparent 3rd-Party Authentication with Application Mobility for 5G Mobile Edge Computing. In Proceedings of the European Conference on Networks and Communications (EuCNC): Network Softwarisation (NET), Dubrovnik, Croatia, 15–18 June 2020.
11. Fotiou, N.; Siris, V.A.; Polyzos, G.C. Capability-based access control for multi-tenant systems using OAuth 2.0 and Verifiable Credentials. *arXiv* **2021**, arXiv:2104.11515.

12. Li, C.-Y.; Lin, Y.-D.; Lai, Y.-C.; Chien, H.-T.; Huang, Y.-S.; Huang, P.-H.; Liu, H.-Y. Transparent AAA Security Design for Low-Latency MEC-Integrated Cellular Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3231–3243. [CrossRef]
13. Kaur, K.; Garg, S.; Kaddoum, G.; Guizani, M.; Jayakody, D.N.K. A Lightweight and Privacy-Preserving Authentication Protocol for Mobile Edge Computing. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
14. Li, Y.; Cheng, Q.; Liu, X.; Li, X. A Secure Anonymous Identity-Based Scheme in New Authentication Architecture for Mobile Edge Computing. *IEEE Syst. J.* **2021**, *15*, 935–946. [CrossRef]
15. Lee, J.; Kim, D.; Park, J.; Park, H. A Multi-Server Authentication Protocol Achieving Privacy Protection and Traceability for 5G Mobile Edge Computing. In Proceedings of the IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–12 January 2021; pp. 1–4.
16. Behrad, S.; Bertin, E.; Tuffin, S.; Crespi, N. A new scalable authentication and access control mechanism for 5G-based IoT. *Futur. Gener. Comput. Syst.* **2020**, *108*, 46–61. [CrossRef]
17. Qiu, Q.; Liu, S.; Xu, S.; Yu, S. Study on Security and Privacy in 5G-Enabled Applications. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 8856683. [CrossRef]
18. Kotulski, Z.; Niewolski, W.; Nowak, T.W.; Sepczuk, M. New Security Architecture of Access Control in 5G MEC. In *Communications in Computer and Information Science*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 77–91.
19. Ji, X.; Huang, K.; Jin, L.; Tang, H.; Liu, C.; Zhong, Z.; You, W.; Xu, X.; Zhao, H.; Wu, J.; et al. Overview of 5G security technology. *Sci. China Inf. Sci.* **2018**, *61*, 081301. [CrossRef]
20. *5G; Security Architecture and Procedures for 5G System*; ETSI TS 133 501 V16.5.0; ETSI: Sophia Antipolis, France, 2021. Available online: https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/16.05.00_60/ts_133501v160500p.pdf (accessed on 23 May 2021).
21. *JSON Web Signature (JWS)*; RFC 7515; IETF: Fremont, CA, USA, 2015. Available online: https://tools.ietf.org/html/rfc7515 (accessed on 23 May 2021).
22. *JSON Web Encryption (JWE)*; RFC 7516; IETF: Fremont, CA, USA, 2015. Available online: https://tools.ietf.org/html/rfc7516 (accessed on 23 May 2021).
23. *JSON Web Key (JWK)*; RFC 7517; IETF: Fremont, CA, USA, 2015. Available online: https://tools.ietf.org/html/rfc7517 (accessed on 23 May 2021).
24. *JSON Web Algorithms (JWA)*; RFC 7518; IETF: Fremont, CA, USA, 2015. Available online: https://tools.ietf.org/html/rfc7518 (accessed on 23 May 2021).
25. *JSON Web Token (JWT)*; RFC 7519; IETF: Fremont, CA, USA, 2015. Available online: https://tools.ietf.org/html/rfc7519 (accessed on 23 May 2021).
26. *Functional Architecture and Information Flows to Support Common API Framework for 3GPP Northbound APIs*. 3GPP TS 23.222 V17.4.0; The 3rd Generation Partnership Project. 2021. Available online: https://www.3gpp.org/ftp/Specs/archive/23_series/23.222/23222-h40.zip (accessed on 23 May 2021).
27. Kotulski, Z.; Nowak, T.W.; Sepczuk, M.; Tunia, M.; Artych, R.; Bocianiak, K.; Osko, T.; Wary, J.-P. Towards constructive approach to end-to-end slice isolation in 5G networks. *EURASIP J. Inf. Secur.* **2018**, *2018*, 2. [CrossRef]
28. Hu, V.C.; Kuhn, D.R.; Ferraiolo, D.F.; Voas, J. *Attribute Based Access Control*; NIST SP 1800-3, Second Draft; NIST: Gaithersburg, MD, USA, 2017.
29. Brik, B.; Frangoudis, P.A.; Ksentini, A. Service-Oriented MEC Applications Placement in a Federated Edge Cloud Architecture. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
30. Grassi, P.A.; Garcia, M.E.; Fenton, J.L. *Digital Identity Guidelines*; NIST SP 800-63-3; NIST: Gaithersburg, MD, USA, 2017. Available online: https://csrc.nist.gov/csrc/media/publications/sp/800-63/3/draft/documents/sp800-63-3-draft.pdf (accessed on 23 May 2021).
31. *Multi-Access Edge Computing (MEC); Phase 2: Use Cases and Requirements*; ETSI GS MEC 002 V2.1.1; ETSI: Sophia Antipolis, France, 2018. Available online: https://www.etsi.org/deliver/etsi_gs/mec/001_099/002/02.01.01_60/gs_mec002v020101p.pdf (accessed on 23 May 2021).
32. *Multi-Access Edge Computing (MEC). MEC 5G Integration*; ETSI GR MEC 031 V2.1.1; ETSI: Sophia Antipolis, France, 2020. Available online: https://www.etsi.org/deliver/etsi_gr/MEC/001_099/031/02.01.01_60/gr_MEC031v020101p.pdf (accessed on 23 May 2021).
33. Behrad, S.; Bertin, E.; Crespi, N. A survey on authentication and access control for mobile networks: From 4G to 5G. *Ann. Telecommun.* **2019**, *74*, 593–603. [CrossRef]
34. *5G; Procedures for the 5G System (5GS)*; ETSI TS 123 501 V16.5.0; ETSI: Sophia Antipolis, France, 2020. Available online: https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/16.05.00_60/ts_123502v160500p.pdf (accessed on 23 May 2021).
35. *Procedures for the 5G System*; 3GPP TS 23.502 V17.0.0; ETSI: Sophia Antipolis, France, 2021. Available online: https://www.3gpp.org/ftp/Specs/archive/23_series/23.502/23502-h00.zip (accessed on 23 May 2021).

36. Hu, V.C.; Ferraiolo, D.; Kuhn, R.; Schnitzer, A.; Sandlin, K.; Miller, R.; Scarfone, K. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*; NIST SP 800-162; NIST: Gaithersburg, MD, USA, 2014. Available online: https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf (accessed on 23 May 2021).
37. Ali, B.; Gregory, M.A.; Li, S. Multi-Access Edge Computing Architecture, Data Security and Privacy: A Review. *IEEE Access* **2021**, *9*, 18706–18721. [CrossRef]
38. Bhatt, S.; Tawalbeh, L.A.; Chhetri, P.; Bhatt, P. Authorizations in Cloud-Based Internet of Things: Current Trends and Use Cases. In Proceedings of the Fourth International Conference on Fog and Mobile Edge Computing (FMEC), Rome, Italy, 10–13 June 2019; pp. 241–246.
39. Zhu, Y.; Gan, G.; Guo, R.; Huang, D.; Ruiqi, G. PHE: An efficient traitor tracing and revocation for encrypted file syncing-and-sharing in cloud. *IEEE Trans. Cloud Comput.* **2018**, *6*, 1110–1124. [CrossRef]
40. Zhang, Q.; Wang, S.; Zhang, D.; Wang, J.; Zhang, Y. Time and Attribute Based Dual Access Control and Data Integrity Verifiable Scheme in Cloud Computing Applications. *IEEE Access* **2019**, *7*, 137594–137607. [CrossRef]
41. Hao, J.; Liu, J.; Wang, H.; Liu, L.; Xian, M.; Shen, X. Efficient Attribute-Based Access Control with Authorized Search in Cloud Storage. *IEEE Access* **2019**, *7*, 182772–182783. [CrossRef]
42. Sciancalepore, S.; Piro, G.; Caldarola, D.; Boggia, G.; Bianchi, G. OAuth-IoT: An access control framework for the Internet of Things based on open standards. In Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017; pp. 676–681.
43. Gupta, D.; Bhatt, S.; Gupta, M.; Kayode, O.; Tosun, A.S. Access Control Model for Google Cloud IoT. In Proceedings of the 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), Baltimore, MD, USA, 25–27 May 2020; pp. 198–208.
44. Krishna, S. JSON Web Token (JWT) based client authentication in Message Queuing Telemetry Transport (MQTT). *arXiv* **2019**, arXiv:1903.02895.
45. Ahmed, S.; Mahmood, Q. An authentication based scheme for applications using JSON web token. In Proceedings of the 22nd International Multitopic Conference (INMIC), Islamabad, Pakistan, 29–30 November 2019; pp. 1–6.
46. Ethelbert, O.; Moghaddam, F.F.; Wieder, P.; Yahyapour, R. A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications. In Proceedings of the IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, Czech Republic, 21–23 August 2017; pp. 47–53.
47. Alkhulaifi, A.; El-Alfy, E.M. Exploring Lattice-based Post-Quantum Signature for JWT Authentication: Review and Case Study. In Proceedings of the IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 25–28 May 2020; pp. 1–5.
48. Rahmatulloh, A.; Gunawan, R.; Nursuwars, F.M.S. Performance comparison of signed algorithms on JSON Web Token. *IOP Conf. Ser. Mater. Sci. Eng.* **2019**, *550*, 012023. [CrossRef]
49. Talukdar, T.; Batra, G.; Vaidya, J.; Atluri, V.; Sural, S. Efficient Bottom-Up Mining of Attribute Based Access Control Policies. In Proceedings of the IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), San Jose, CA, USA, 15–17 October 2017; pp. 339–348.
50. Aldy, P.A.; Alam, R.; Nur, A. Stateless Authentication with JSON Web Tokens using RSA-512 Algorithm. *J. Infotel* **2019**, *11*, 36–42. [CrossRef]
51. Guo, L.; Yang, X.; Yau, W.C. TABE-DAC: Efficient Traceable Attribute-Based Encryption Scheme with Dynamic Access Control Based on Blockchain. *IEEE Access* **2021**, *9*, 8479–8490. [CrossRef]
52. Zhu, Y.; Yu, R.; Ma, D.; Chu, W.C.-C. Cryptographic Attribute-Based Access Control (ABAC) for Secure Decision Making of Dynamic Policy With Multiauthority Attribute Tokens. *IEEE Tran. Reliab.* **2019**, *68*, 1330–1346. [CrossRef]
53. Chang, D.; Sun, W.; Yang, Y.; Wang, T. An E-ABAC-Based SDN Access Control Method. In Proceedings of the 6th International Conference on Information Science and Control Engineering (ICISCE), Shanghai, China, 20–22 December 2019; pp. 668–672.
54. Pencheva, E.; Asenov, I.; Atanasov, I.; Trifonov, D.V. Programmability of Policy Control at the Edge of the Mobile Network. In Proceedings of the International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, Canada, 20–22 October 2020; pp. 1–6. [CrossRef]
55. Nife, F.N.; Kotulski, Z. Application-Aware Firewall Mechanism for Software Defined Networks. *J. Netw. Syst. Manag.* **2020**, *28*, 605–626. [CrossRef]
56. Ayache, M.; Gawanmeh A.; Al-Karaki, J.N. XBAC: A Unified Access Control Model for Heterogeneous Multi-Tenancy Cloud Environments. In Proceedings of the 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 1872–1878.
57. JWCrypto Documentation, Rev. 4e08b661. Available online: https://jwcrypto.readthedocs.io/ (accessed on 23 May 2021).
58. Cryptography Documentation. Available online: https://cryptography.io/en/latest/hazmat/backends/openssl/ (accessed on 23 May 2021).
59. OpenSSL Documentation. "Frequently Asked Questions". Available online: https://www.openssl.org/docs/faq.html#MISC (accessed on 23 May 2021).

60. AbdAllah, E.G.; Kuang, Y.R.; Huang, Ch. Advanced Encryption Standard New Instructions (AES-NI) Analysis: Security, Performance, and Power Consumption. In Proceedings of the 2020 12th International Conference on Computer and Automation Engineering (ICCAE), Sydney, Australia, 14–16 February 2020; pp. 167–172.
61. Kotulski, Z.; Szczepinski, W. *Error Analysis with Applications in Engineering*; Springer: Dordrecht, The Netherlands, 2010.
62. *Use of Public Standards for the Secure Sharing of Information among National Security Systems*; CNSS Advisory Memorandum, Information Assurance 02-15; July 2015. Available online: https://cryptome.org/2015/08/CNSS_Advisory_Memo_02-15.pdf (accessed on 23 May 2021).
63. Srinivasa, R.; Naidu, N.K.S.; Maheshwari, S.; Bharathi, C.; Kumar, A.R.H. Minimizing Latency for 5G Multimedia and V2X Applications using Mobile Edge Computing. In Proceedings of the 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), Jaipur, India, 28–29 September 2019; pp. 213–217.