Quantum-safe Edge Applications: How to Secure Computation in Distributed Computing Systems

Claudio Cicconetti
IIT-CNR
Pisa, Italy

Dario Sabella
Intel Corporation
Turin, Italy

Pietro Noviello
Exprivia S.p.A.
Molfetta, Italy

Gennaro Davide Paduanelli
Exprivia S.p.A.
Molfetta, Italy

Abstract—The advent of distributed computing systems will offer great flexibility for application workloads, while also imposing more attention to security, where the future advent and adoption of quantum technology can introduce new security threats. For this reason, the Multi-access Edge Computing (MEC) working group at ETSI has recently started delving into security aspects, especially motivated by the upcoming reality of the MEC federation, which involves services made of application instances belonging to different systems (thus, different trust domains). On the other side, Quantum Key Distribution (QKD) can help strengthen the level of security by enabling the exchange of secure keys through an unconditionally secure protocol, e.g., to secure communication between REST clients and servers in distributed computing systems at the edge. In this paper, we propose a technical solution to achieve this goal, building on standard specifications, namely ETSI MEC and ETSI QKD, and discussing the gaps and limitations of current technology, which hamper full-fledged in-field deployment and mass adoption. Furthermore, we provide our look-ahead view on the future of secure distributed computing through the enticing option of federating edge computing domains.

Index Terms—quantum key distribution, QKD, edge computing, ETSI MEC, ETSI QKD

I. INTRODUCTION

Distributed computing scenarios offer a variety of computing partitioning, with the advantage of potentially optimizing the distribution of the workload to minimize costs and maximize performance [1]. In this context, standardization of the edge infrastructure ensures interoperability, where *federation*, as defined by ETSI MEC [2], can be a convenient framework to enable this kind of heterogeneous deployments, as shown in Figure 1. Securing communications in those complex scenarios is however critical, and quantum cryptography and Quantum Key Distribution (QKD) can help in strengthening the level of security of these environments.

QKD is the most mature among the Quantum Information Technologies, which are all advancing at a fast pace to revolutionize both computing, through a speed-up of computation to solve problems that are intractable with classical (i.e., non-quantum) computers [3], and communication, with the deployment of the Quantum Internet [4]. However, despite some ongoing efforts to build quantum networks at a geographical scale (e.g., the EuroQCI initiative in EU[1]), the goal of building
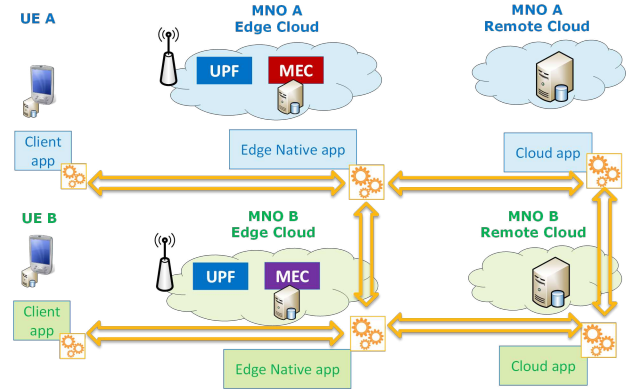

Fig. 1. Distributed computing scenarios

a global infrastructure in the next years is too ambitious, mainly because of two reasons. First, quantum communications rely on the encoding of quantum states in single photons, which makes the transmission very fragile, especially at long distances. Second, quantum bits cannot be copied/amplified, due to the "no-cloning" theorem of quantum mechanics, which makes the manufacturing of so-called quantum repeaters overly complex [5]. For this reason, it is generally assumed that, at least in the near future, quantum networks will be limited to Quantum Metropolitan Area Networks (QMANs), which makes them an excellent candidate for being used in tandem with edge computing, possibly with the help of Trusted Nodes (TNs) [6]. A TN is a device that interconnects two QKD devices and exchanges secret keys with both of them to create a logical tunnel for the secure end-to-end exchange of keys, which however is only as secure as the TN itself.

The remainder of this paper is structured as follows. In Section II we illustrate two example use cases that motivate the need for secure edge computing. The interworking of ETSI MEC and QKD to achieve this goal is described in detail in Section III, which is followed by a high-level look ahead on edge computing federation in Section IV. Section V concludes the paper.

II. MOTIVATING USE CASES

A. Cybersecurity in Healthcare

Professional healthcare is undergoing major advances thanks to the broader adoption of Information and Communication Technologies (ICT), especially through the use
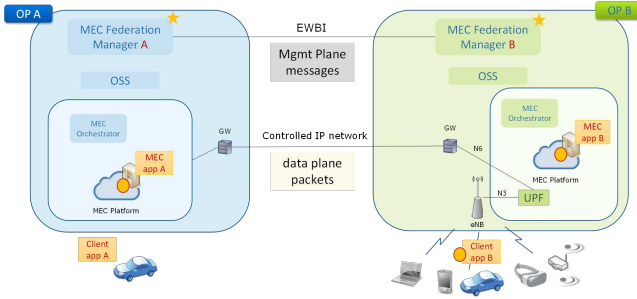
---

[1] https://digital-strategy.ec.europa.eu/en/policies/european-quantum-communication-infrastructure-euroqci

Fig. 2. Exemplary deployment of a MEC federation

Fig. 3. High-level interworking between ETSI MEC and QKD.

of Artificial Intelligence (AI) algorithms to help doctors in the diagnosis phase, for emergency and evaluation of triage, and for continuously monitoring patients. Hospitals, sanitaria, and long-term care facilities are increasing the adoption of body/room sensors for the collection of vital parameters, which can then be used as input for inference models based on Machine Learning (ML)-trained Neural Networks (NNs). The latter can be very computation-intensive and, thus, require high-performance computing infrastructures that are not often available on-site. Such data are highly confidential and protected by many national regulations (e.g., General Data Protection Regulation (GDPR) in the EU member states). This makes a strong case for using QKD and edge computing [7]: the security brought by the former is reinforced by the latter, which keeps the remote interactions as close as possible to where the data are generated/consumed. In this use case, the administrative boundaries (or security perimeters), both hosting QKD devices interconnected through a quantum network, would be the healthcare facilities on the user side, and the edge computing infrastructure on the telco operator side.

B. Cybersecurity in Automotive

Automotive and smart city scenarios in real life are characterized by complex environments with multiple mobile network operators, vendors, and integrators. From an application perspective, these environments are populated by several types of workloads, i.e. client applications, MEC applications and backend applications [8]. In this context, an automotive scenario can be realized with the multi-stakeholder architecture depicted in Figure 2: the figure also exemplifies one of the key use cases for the MEC Federation, where multiple OP instances (as defined by GSMA OPG [9]) are connected via EWBI (East-West Bound Interface) Application Programming Interfaces (APIs), for control plane traffic [10], and via a controlled IP network, for the data plane traffic. The improved security with QKD networks/technologies would increase acceptance, a critical barrier to market penetration for this use case. Due to the mobile nature of vehicular users, QKD devices cannot be co-located with individual terminals; instead, the QKD devices, interconnected to the quantum network via a fiber optic infrastructure, can be deployed in the security perimeter of the vehicular network, e.g., on Road Side Units (RSUs).
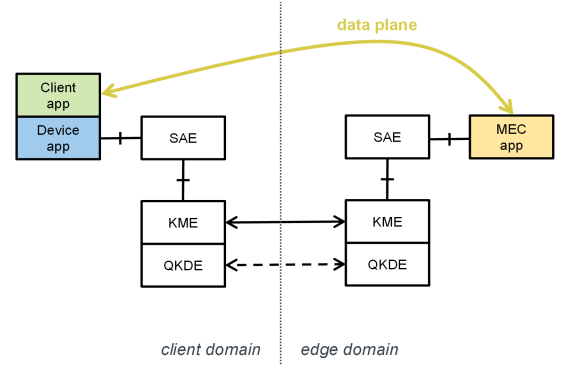
III. INTERWORKING OF MEC AND QKD

At a high level, the proposed solution to integrate QKD with an edge domain, where resources are deployed and managed through ETSI MEC, is illustrated in Figure 3. The diagram shows a client and an edge domain, which are interconnected by both classical (5G and beyond) and quantum networks. By design, the application on the client terminal (*client app*) is transparent to both ETSI MEC and ETSI QKD procedures: this enables a gradual transition of customer applications towards the final goal and reduces the business risks related to the possible emergence of future alternative standards/frameworks. Using ETSI MEC terminology, the client app interacts with a *device app*, which is aware of the ETSI MEC procedures and, in particular, supports the client-side protocol of the Mx2 interface defined in ETSI MEC 016 [12]. The counterpart of the client app in the edge domain is called *MEC app* and, in general, it is provided by the service provider to be run in the edge computing infrastructure as a partial or full substitute for cloud back-end services. The device app exploits two capabilities offered by the ETSI MEC, both available through a component called Life Cycle Management Proxy (LCMP) (not shown in this diagram): i) discovery, i.e., the device app may retrieve the list of MEC apps available on the edge domain, and ii) context creation, i.e., the device app can create an application context which, until its termination, will ensure that resources in the edge domain will be reserved and configured for the specific MEC app to which the context is attached.

Moving to the QKD segment, there are three main entities. First, the *QKD Entity (QKDE)* is the QKD device, that is the termination of the underlying QKD network. Through the continuous generation of locally entangled photons (on one side) and the measurement (on the other side), a pair of QKDEs can establish a stream of secret material that cannot be spoofed. The *Key Management Entity (KME)* is the component that combines such secret material in blocks and keys to make them available to the upper layers, also through a chain of intermediate TNs to extend beyond a local point-to-point connection between QKDEs [13]. According to the ETSI QKD specifications, the KMEs should cooperate to enable the discovery of available QKDEs, agree on the protocol for

combining keys in a multi-hop fashion, and perform other management functions, but the specific procedures and interfaces are left unspecified. On the other hand, the application API is defined in the ETSI QKD 014 [11], which defines a REST protocol to be used by the applications to retrieve security key from a *Secure Application Entity (SAE)*, which in turn interacts with a local KME. The SAE, KME, and QKDE should all reside within the same security perimeter to avoid the secret keys being compromised.

Once the ETSI MEC application context has been created, through the `Mx2` interface, the edge computing resources have been provisioned, and a security context has been created using the ETSI QKD 014 interface, the client and MEC apps can interact with one another by exchanging, through the classical network, encrypted messages that they only can decrypt with QKD-secure keys, as shown in the "data plane" bidirectional arrow in Figure 3. In the remainder of this section, we provide additional details following a prototype implementation [14].

In the following, we assume that the client and MEC app interact through HTTP request/response messages. This pattern is typical, for instance, of the Function as a Service (FaaS) abstraction, often used in combination with serverless computing, which is a mature and very popular cloud-native technology [15]. With FaaS, the service is broken down into elementary pieces that are invoked as functions in an edge/cloud serverless platform, either directly by the client or from other functions in a chain (or Directed Acyclic Graph (DAG)) of consecutive invocations, until the final result is returned to the client. In this paper, we are interested only in encrypting the first and last step, i.e., the invocation of the client that triggers the execution of the function(s) and the last return value from the platform, since both may contain customer confidential data. The intermediate invocations, if any, all happen within the edge domain, which is assumed to be within the same security perimeter, and therefore QKD-secure encryption is not necessary. Furthermore, we are not concerned with the required exchanges within the QKD network to build secure keys across a chain of TNs, which do not affect the focus of our contribution and is a topic under study within the scientific literature (e.g., [16]) and standardization bodies such as the ITU-T Focus Group on Quantum Information Technology for Networks (FG-QIT4N). Therefore, without loss of generality, below we represent the QKD network as a point-to-point link.

The ETSI MEC components of the architecture are available as open-source code, programmed in Rust, under a permissive MIT license on GitHub[2].

A. Detailed Software Architecture

A detailed software architecture of the proposed solution is illustrated in Figure 4, which shows additional components than those in the high-level view in Figure 3. Specifically, in the client domain, in addition to the client app and device app, we find a *reverse proxy*, which captures the HTTP requests/responses between the client app and the MEC app,

to keep the former transparent to ETSI MEC/QKD procedures, and, the *client key store*, which is an in-memory database that keeps data required for the encryption/decryption and context management. Both the reverse proxy and the client key store can serve multiple clients within the same client domain, which may cover, for instance, a factory or a hospital served through a 5G private network and with a QKD node. In the ETSI MEC domain, Figure 4 shows the LCMP, two *MEC hosts*, which are the physical/virtual edge computing nodes dedicated by the telco operator to hosting MEC applications, the *MEC host key store*, which is an in-memory database storing the keys used by the MEC app for encryption/decryption, and the *MEC Orchestrator (MEO)*, which is the entity within an ETSI MEC domain in charge of managing the deployment and run-time optimization of the MEC apps.

In step **1** the user authenticates on their terminal via the local mobile app, which triggers the request for a service creation, issued by a REST command with HTTP and captured by the reverse proxy (**2a**). The latter then directs an app initialization to the device app (**2b**), which is the entry point towards the ETSI MEC, also shared with other clients in the same domain. The device app first performs an application look-up via the discovery features of the remote LCMP (**3a**) to ensure that the type of MEC app requested is offered by a service provider hosted by the ETSI MEC domain, with matching version and capabilities as requested by the device app. If this is the case, then the device app proceeds to request the creation of an app context to the LCMP (**3b**). The exchanges between the device app and the LCMP are not subject to high-security constraints since no confidential data are exchanged, but traditional security means, such as Transport Layer Security (TLS) can optionally be used.

The LCMP then informs the MEO of the request to create a new app context (**4**), which triggers a resource orchestration decision at the MEO. It looks for a suitable MEC app already deployed that can serve the new app context and, if not found, deploys a new MEC app on one of the MEC hosts with suitable characteristics, depending on the available resources and their locations, and the current status of utilization as obtained through the observability framework within the ETSI MEC domain. In the literature, the problem of optimal allocation of resources in edge computing has been broadly investigated, typically to maximize resource utilization or minimize energy or latency (e.g., [17], [18]). Once the MEC app is deployed, the MEO can return the end-point to be used by the client app, through a Uniform Resource Identifier (URI), to the LCMP (**6**), which embeds it into the app context creation response returned to the device app (**7**).

At this point, the device app knows the end-point of the MEC app, which is cached in the client key store (**8**) and passes the original request from the client, with confidential data, to the SAE app. The latter establishes a security context creation with the corresponding SAE app in the MEC host where the MEC app is deployed. This procedure is illustrated separately later and it involves the QKD network underneath. The security context creation concludes with the SAE app
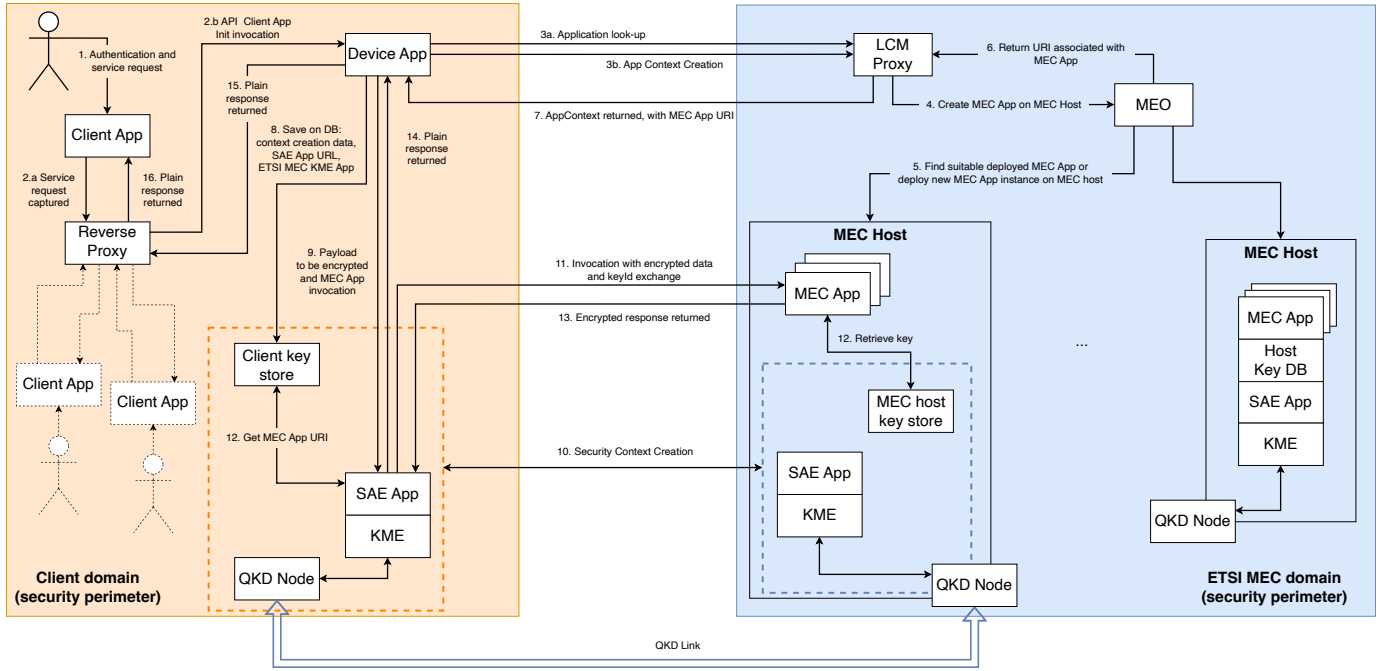
Fig. 4. Detailed software architecture of an ETSI MEC computing infrastructure providing clients with services, also including encryption of private data with keys made available by a QKD network through an ETSI QKD 014 API. The security context creation procedure (step **10**) is shown separately in Figure 5.

having a secret symmetric key with which it can encrypt the confidential client data before invoking the REST command on the MEC app listening at the given URI (**11**). For instance, with serverless computing, such a command is a function invocation and the client data are passed as function arguments. Upon receiving the invocation, the MEC app can retrieve its symmetric key from the MEC host key store (**12**), which plays the same role as that used by the SAE app for encrypting the client data, to decrypt and then consume the data. It is worth noting that retrieving the key requires the identifier of the key (keyId), which was obtained during the security context creation procedure and passed by the SAE app as part of the invocation. Once processing is complete, which may involve also the interaction with other MEC apps, not shown in the diagram, the final result is encrypted by the entry-point MEC app with the same key used for decrypting and returned to the SAE app (**13**). The latter can decrypt the response and forward it to the device app (**14**), which in turn passes it to the reverse proxy (**15**) and, eventually, reaches the client app (**16**).

B. Security context creation

We now focus on the security context creation (step **10** in Figure 4), which is illustrated through the sequence diagram in Figure 5. In both domains, the three actors are the SAE App, the KME, and a key store. The latter is used to cache keys and reuse them for multiple consecutive exchanges, as the KME, according to the ETSI QKD specifications, is not allowed to store secret material once it has been consumed by the SAE app. Reusing the same key introduces a security threat since, theoretically, this makes it possible for an attacker to break the secret, but it allows more efficient use of the QKD resources, which are rather scarce with current technology (in the order

of kbit/s [19]). The trade-off between security and efficiency in the selection of the key refresh rate depends on the specific characteristics and security targets of the application.

The procedure starts with the client advertising the set of known protocols, called CypherSuites in the diagram, to encrypt data to the MEC app, which selects a match (if any) and returns it. This step determines, along with the algorithm, the length of the key to be used (keyLength). The next step is done by the client, which uses the ETSI QKD 014 API to retrieve a key from its KME, also specifying an identifier of its peer, i.e., the MEC app. Depending on the QKD network topology, the KME may have to negotiate with multiple KMEs for an end-to-end key to be created, but as already mentioned this step is not discussed in this paper, since it is transparent from the point of view of the SAE apps. The KME then returns the client app a key with the given length and an identifier (keyId), and they are both saved in the client key store for later use.

The client app then sends to the MEC app the identifier of the client app, the keyId, and the CypherSuite. With the identifier and the keyId, the MEC app can query its MEC KME via the ETSI QKD 014 API and retrieve the associated key, which is then cached in the key store of the MEC host for later use, similarly to how was done by the client.

C. Discussion

The use of QKD is subject to high deployment cost, in terms of both fiber optic cores that must be dedicated to QKD links and new equipment to be installed. Therefore, it is tempting to evaluate the use of Post Quantum Cryptography (PQC) as a tout court alternative, as the latter merely requires software upgrades. On the one hand, we can foresee that
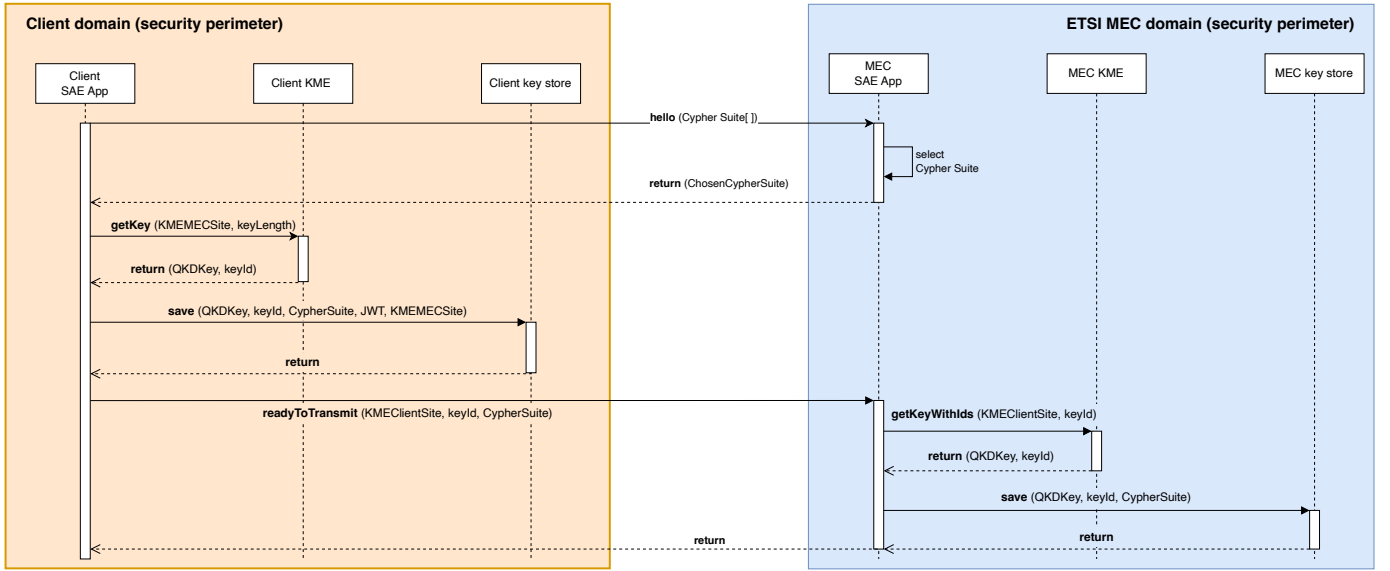
Fig. 5. Security context creation procedure within the proposed software architecture (step **10** in Figure 4).

technology development will improve efficiency and reduce manufacturing costs of quantum devices, thus making QKD more cost-effective. On the other hand, PQC, while being a matter subject to intense research, will always be dependent on the computational power ascribed to the adversary, whereas QKD can be mathematically proven to be secure, independent of the resources of the adversary, being an Information Theory Secure (ITS) primitive [20].

Another deployment issue is that end-to-end security requires a QKD device to be installed in the security perimeter of the customer, which creates a new form of *last mile* problem: if the customer cannot connect directly to the fiber-optic quantum network, then it is not possible to extend coverage through a copper-based or wireless 5G-like access network since they do not allow, at least today, the transmission of single photons encoded with a quantum state. This is particularly relevant to scenarios where user terminals are mobile, such as in an automotive use case (Section II-B). One possibility under study is the use of free space through satellites, which is promising with both geo-stationary [21] and low-orbit [22] scenarios, though the related technology is lagging behind terrestrial fiber optic based quantum communications [23].

Finally, in the solution illustrated above, the QKD network and the edge computing domain take independent resource management decisions: the former on the path to follow between the client and edge domains, the latter on which MEC host to use to deploy the MEC app. This works well for demonstration purposes or with low adoption of the technology, but, in a fully grown scenario, it may lead to inefficient utilization of resources. This can be overcome by a run-time management interaction between the QKD and edge computing operators, which is a topic understudied (despite some initial attempts, such as [24]) and, to the best of our knowledge, not yet covered by standardization bodies.

IV. LOOKING AHEAD

The federation of multiple infrastructures is a thriving topic in the edge-cloud ecosystem, which multiplies the importance of security. Figure 6 shows a possible communication between application instances with MEC federation, with proper security enabled by QKD in both MEC systems and the respective client domains. However, even if the high-level concept is quite straightforward, proper standardization work is needed to allow actual implementations to fully exploit the benefits of QKD in MEC deployments. In fact, according to ETSI GS QKD 014, the communication between applications in different security perimeters is assumed to be insecure, as performed via so-called *public channels*. This is consistent with the assumption in MEC systems that data plane connectivity between MEC apps is generically insecure. Therefore, QKD-enabled secure communication between MEC apps with federation would be utterly needed.

As a starting point, ETSI QKD has already defined suitable QKD Key Servers [25], which, at least in principle, can be conveniently deployed, e.g., co-located with the MEC Federator (MEF) or MEO, or stand-alone. But many gaps need to be filled to achieve interoperability along the following directions, from the point of view of ETSI MEC:

1. definition of signaling to allow protocol support for QKD-based authentication in a MEC federation → impact on authentication mechanisms defined in **ETSI MEC 009**;
2. extension of the application profile to support exchanges between SAEs related to multiple MEC app instances → relevant for future releases of **ETSI MEC 011**, for example by properly enhancing the AppInfo data type by including an indication of the availability of a QKD Key Server in the system;
3. support in the MEC federators and the related flows → enhanced Federation Enablement APIs in **ETSI MEC 040**.
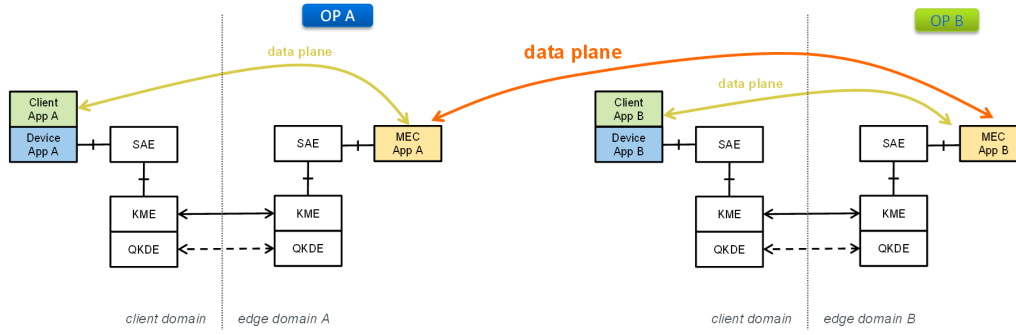
Fig. 6. QKD-secure communication in MEC federation.

It is even possible to conceive ETSI MEC service APIs specifically to support secure communication between MEC applications in a federation, or adopting reference software implementations from relevant open-source communities.

V. CONCLUSIONS

In this paper, we have discussed a scenario for enabling QKD-secure data exchange in cloud-native applications deployed at the edge. As a reference, we have considered a computing infrastructure based on the ETSI MEC architecture, and we have assumed that the quantum devices offer ETSI QKD 014 APIs. Furthermore, we have discussed some key threats and opportunities for the future growth of the technology, towards a mass deployment of secure distributed computing.

ACKNOWLEDGMENT

REFERENCES

[1] Chen, Q., Guo, S., Wang, K., Xu, W., Li, J., Cai, Z., Gao, H. & Zomaya, A. Towards Real-Time Inference Offloading with Distributed Edge Computing: the Framework and Algorithms. *IEEE Transactions On Mobile Computing*. pp. 1-18 (2023).
[2] ETSI, "ETSI GS MEC 003 V3.1.1 (2022-03). Multi-access Edge Computing (MEC); Framework and Reference Architecture,", 2022.
[3] Quantum Technology and Application Consortium — QUTAC, "Industry quantum computing applications," *EPJ Quantum Technology*, vol. 8, no. 1, p. 25, Dec. 2021.
[4] L. Gyongyosi and S. Imre, "Advances in the quantum internet," *Communications of the ACM*, vol. 65, no. 8, pp. 52–63, 2022.
[5] Rakonjac, J., Corrielli, G., Lago-Rivera, D., Seri, A., Mazzera, M., Grandi, S., Osellame, R. & Riedmatten, H. Storage and analysis of light-matter entanglement in a fiber-integrated system. *Science Advances*. **8**, eabn3919 (2022,7).
[6] Y. Cao, Y. Zhao, Q. Wang, J. Zhang, S. X. Ng, and L. Hanzo, "The Evolution of Quantum Key Distribution Networks: On the Road to the Qinternet," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 839–894, 2022.
[7] Ur Rasool, R., Ahmad, H., Rafique, W., Qayyum, A., Qadir, J. & Anwar, Z. Quantum Computing for Healthcare: A Review. *Future Internet*. **15**, 94 (2023,3).

[8] 5GAA Automotive Association, Working Group gMEC4AUTO. "MEC System Interoperability and Test Framework" (v0.9.1), 2023.
[9] GSMA. "Operator Platform Telco Edge Requirements" (v2.0), 2022.
[10] ETSI, "ETSI GS MEC 040 V3.1.1 (2023-02). Multi-access Edge Computing (MEC); Federation enablement APIs ,"., 2022.
[11] ETSI, "ETSI GS QKD 014 V1.1.1 (2019-02). Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API," 2019.
[12] ETSI, "ETSI GS MEC 016 V2.2.1 (2020-04). Multi-access Edge Computing (MEC); Device application interface ," 2020.
[13] Tysowski, P., Ling, X., Lütkenhaus, N. & Mosca, M. The Engineering of a Scalable Multi-Site Communications System Utilizing Quantum Key Distribution (QKD). *Quantum Science And Technology*. (2017,11).
[14] Cicconetti, C., Conti, M., Lella, E., Noviello, P., Paduanelli, G., Passarella, A. & Storelli, E. A Prototype for QKD-secure Serverless Computing with ETSI MEC. *2023 IEEE International Conference On Smart Computing (SMARTCOMP)*. pp. 189-190 (2023,6).
[15] A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. D. A. Popa, J. E. Gonzalez, I. O. N. Stoica, and D. A. Patterson, "What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing," *Communications of the ACM*, vol. 64, no. 5, pp. 76–84, 2021.
[16] Kong, P. Challenges of Routing in Quantum Key Distribution Networks with Trusted Nodes for Key Relaying. *IEEE Communications Magazine*. pp. 1-7 (2023).
[17] Younis, A., Maheshwari, S. & Pompili, D. Energy-Latency Computation Offloading and Approximate Computing in Mobile-Edge Computing Networks. *IEEE Transactions On Network And Service Management*. pp. 1-1 (2024).
[18] Dai, X., Xiao, Z., Jiang, H., Lei, M., Min, G., Liu, J. & Dustdar, S. Offloading Dependent Tasks in Edge Computing with Unknown System-side Information. *IEEE Transactions On Services Computing*. pp. 1-15 (2023).
[19] D. Bacco, I. Vagniluca, B. Da Lio, N. Biagi, A. Della Frera, D. Calonico, C. Toninelli, F. S. Cataliotti, M. Bellini, L. K. Oxenløwe, and A. Zavatta, "Field trial of a three-state quantum key distribution scheme in the Florence metropolitan area," *EPJ Quantum Technology*, vol. 6, no. 1, p. 5, Dec. 2019.
[20] Scarani, V., Bechmann-Pasquinucci, H., Cerf, N., Dušek, M., Lütkenhaus, N. & Peev, M. "The security of practical quantum key distribution". *Reviews Of Modern Physics*. **81**, 1301-1350 (2009,9).
[21] J. Yin *et al.*, "Satellite-based entanglement distribution over 1200 kilometers," *Science*, vol. 356, no. 6343, pp. 1140–1144, Jun. 2017.
[22] Gündoğan, M., Sidhu, J., Henderson, V., Mazzarella, L., Wolters, J., Oi, D. & Krutzik, M. Proposal for space-borne quantum memories for global quantum networking. *Npj Quantum Information*. **7**, 128 (2021,12).
[23] Gatto, A., Ferrari, M., Brunero, M., Gagliano, A., Tarable, A., Bodanapu, D., Giorgetti, A., Andriolli, N., Paganelli, R., Strambini, L., Martelli, P. & Martinelli, M. Integration of QKD Technologies in Advanced Optical Networks. *2022 IEEE 15th Workshop On Low Temperature Electronics (WOLTE)*. pp. 1-4 (2022,6).
[24] Cicconetti, C., Conti, M. & Passarella, A. Qkd@Edge: Online Admission Control of Edge Applications with QKD-secured Communications. *2023 IEEE International Conference On Smart Computing (SMARTCOMP)*. pp. 65-73 (2023,6).
[25] ETSI, "ETSI GS QKD 004 V2.2.1 (2020-08). Quantum Key Distribution (QKD); Application Interface," 2020.