Quantum Cryptography Networks Supporting Path Verification in Service Function Chains

A. Aguado, D. R. López, A. Pastor, V. López, J. P. Brito, M. Peev, A. Poppe, and V. Martín,

***Abstract*—Quantum Key Distribution is a physical technology that enables the secure generation of bit streams (secret keys) in two separated locations. This technology is designed to provide a solution for very secure (quantum-safe) key agreement, which is nowadays at a risk due to advances in quantum computing. The recent demonstration of a QKD network in the metropolitan area of Madrid shows how these networks can be deploy in current production infrastructure by following existing networking paradigms, such as Software-Defined Networking (SDN). In particular, a three-node QKD network is implemented on the metropolitan area network using existing infrastructure and coexisting with other data and control services.**

On the other hand, telecommunication networks are drastically changing the way the services are architectured. Users of the operator's infrastructure are moving from traditional connectivity services (e.g. virtual private networks -VPNs-) to a set of interconnected network functions, either physical (PNF) or virtual (VNF), in the shape of service functions chaining (SFC). However, SFC users do not have a method to validate that the traffic flow is appropriately forwarded across the nodes in the network, situation that may lead to very critical security breaches (for example, a security node or a firewall that is bypassed on the chain). This work presents a method for validating ordered proof-of-transit (OPoT) on top of the Madrid Quantum Network. We first provide an general description of the QKD network deployed in Madrid. Then, we describe an existing security protocol for proof-of-transit in packet networks, analysing its issues and vulnerabilities. Finally, this work presents a protocol for alleviating the security breach found in this work and for providing ordered proof-of-transit (OPoT) in SFCs. Finally, an example of the real implementation is shown, where nodes being part of the OPoT scheme are provisioned with QKD-derived keys.

***Index Terms*—Proof of Transit, Quantum key distribution, Network virtualization, Software Defined Networks.**

I. Introduction

A. Aguado, J. P. Brito and V. Martin are with Center for Computational Simulation, Universidad Politecnica de Madrid, Campus Montegancedo, 28660 Boadilla del Monte, Madrid, Spain (e-mail: {a.aguadom, jp.brito, vicente}@fi.upm.es).

D. R. Lopez, A. Pastor and V. Lopez are with Telefonica GCTO, Ronda de la Comunicacion s/n 28050 Madrid, Spain (e-mail: {diego.r.lopez, antonio.pastorperales, victor.lopezalvarez}@telefonica.com).

M. Peev and A. Poppe are with Huawei Technologies Duesseldorf GmbH, Riesstrasse 25, 80992 Munchen, Germany (e-mail: {momtchil.peev, andreas.poppe}@huawei.com).

Manuscript received August 14, 2018.

THE conventional data and control communication channels within the telecommunication networks are under risk due to the advances on computation. Traditional cryptosystems, that tipically use public encryption algorithms (in particular for secret key agreement), are at risk due to developments in computing. More specifically, the evolution of quantum computing can compromise the existing cryptosystems, which base their security in mathematical problems that, while are assumed complex to solve in classical computing, are easy in quantum computers. QKD is immune to this threat [1], [2]. It allows to generate synchronized random bits in two sources that are separated in space, with the additional property that the maximum information leaked out during the QKD process can be upper bounded. This means that the random bits can be used as a secret key. The security of the QKD keys is due to the laws of quantum mechanics and, therefore, independent of any computational assumption and immune to any algorithmic cryptanalysis.

QKD is a new opportunity for operators as it brings an additional physical layer for securing the communications infrastructure. The *Madrid Quantum Network*, presented in [3], [4], is a new QKD network that distinguishes itself in using operational infrastructure and carrier-grade optical devices to transport the quantum channels together with multiple classical channels in a shared quantum-classical infrastructure that is managed and exploited through the SDN and NFV paradigms. The network has been recently deployed in downtown Madrid, across a metro area network. This network acted as the main source for symmetric keys that were used to secure control [5] and data [6], [7] channels, demonstrating how QKD is a suitable technology for securing the infrastructure and services by combining existing key exchange algorithms together with quantum-generated keys in a hybrid quantum and classical security scheme.

Aside from the traditional control and data channels describe above, the network architecture and the services deployed on top are drastically changing, forcing operators to evolve from traditional/legacy, non-scalable networks towards new architectural solutions. This evolution is powered mainly by three different sources, first the development of new hardware devices

and the increased capacity of existing ones. Second, the creation and re-definition of existing control plane and routing protocols (e.g. segment-routing [8], key-flow) and third, the evolution towards software-defined and virtualized networking architectures. This progress (specially SDN [9] and NFV [10]) allows to dynamically allocate network resources per-user/service and on-demand. However, the flexibility brought by the new software networking trends also carry certain associated vulnerabilities and implications. For instance, in a virtualized environment, several functions might be deployed in distributed locations for composing a service function chain (SFC) [11]. Both control and data communications must be appropriately secured, as any attempt to compromise a virtual function or its behaviour can compromise the entire infrastructure.

A wide-spread concern about virtualized network elements is related to traffic attestation. Any network device deployed in a production network must be capable of assessing if a specific traffic flow passes through it and is correctly forwarded. If a node cannot guarantee this capability, it will not be accepted for production deployment. By progressively changing physical network functions (PNFs) by virtual network functions (VNFs), this task becomes harder. As the traffic traverses multiple intermediate nodes (possibly, out of the control of the VNF operator), it could eventually bypass a critical node within the SFC (e.g. a firewall). In order to mitigate this issue, a first-of-a-kind proof-of-transit technique [12] has been developed within the IETF to verify if a packet has traversed all the nodes within a path. By unique or first-of-a-kind we mean that there is no similar approach defined or available in previous studies. The closest solution is also described in the very same document, which implies a nested encryption scheme. This is computationally costly for network elements and requires more data to be transmitted per packet (higher overhead). Other solutions have studied similar approaches using mathematical theorems for networking purposes. An example is described in [13], where authors implemented a routing protocol based on residual number systems (Chinese remainder theorem). Although very innovative, this method does not focus on traffic attestation, as it defines a low-consumption routing method based on modular operations. The PoT approach originally defined in [12], as we further describe in the next section, does not verify order of the followed path, whilst also have some vulnerabilities associated. These weaknesses could potentially be exploited by a given attacker to bypass nodes in the chain, opening security breaches (e.g. the bypass of a firewall or security gateway). Symmetric key encryption can be used as a way to enhance the existing solution by securing and adding further capabilites, as it is discussed below.

In this work, we propose a way to enhance the current PoT scheme proposed in [12], by using symmetric key encryption (potentially, one time pad -OTP- [14], but could be any other algorithm) for ciphering or masking the associated traffic's metadata. Firstly, we provide an overview of the *Madrid Quantum Network* (MQN), a real QKD network deployed in Telefonica Spain's production facilities where the experiments were done. Then we describe the existing technique for implementing PoT, while we analyse the vulnerabilities of the scheme. Our proposal not just solves these vulnerabilies within the current model, but goes beyond by providing order to the proof-of-transit scheme (OPoT) in an clean and easy to implement way. These enhancements have been presented by the authors of this paper in the IETF and are currently being proposed as the way to implement order in the solution presented at the IETF draft, replacing the previous nested encryption proposal. Also, in order to provide a practical demonstration of the highest possible security (secure symmetric keys) and going beyond a typical setting, we have implemented the solution over the MQN. This allows to show how the solution can also integrate well with less standard techniques. Using QKD also allows to demonstrate a quantum-safe technique that will survive in case of major algorithmic threats (e.g. the one posed by quantum computers).

II. The *Madrid Quantum Network* -MQN- field trial

The *Madrid Quantum Network* (MQN) [3], [4] has been recently deployed in a metropolitan area network around Madrid city center. The network has three main locations: Almagro (Telefónicas I+D laboratory shared with Telefónica Spain PoP), Norte and Concepcion (both are Telefónica Spain PoPs). The three nodes and the three connections (three pairs of fibers) are part of Telefónica Spain production network. The QKD systems were developed by Huawei Research Center Germany using continuous variables technology [15], [16], [17] that is more resilient to noise from the classical channels sharing the same fibre with the quantum channel. The QKD network manage the key generation according to SDN principles by switching the quantum states generated by the single transmitter to one of the both receivers. This control by an SDN network was one of the design targets of the *Madrid Quantum Network* testbed and a requirement for quantum/classical network convergence. Almagro hosts the QKD transmitter, while two QKD receivers are distributed to Norte and Concepcion. The connection go across other PoPs, where splices and connectors in patch panels causes additional fiber losses. Some general parameters of the links:

- Almagro-Norte: around 3.9km long with a loss of 6dB and a key generation capability -estimated from

the raw key- above 70kbps, with coexistence of 17 classical channels together with the quantum one in the same (C) band. This limit was due to the number of free slots available in the optical equipment, so the number could be higher.

- Almagro-Concepcion: around 6.4km long with a loss of $\sim$ 7dB. We artificially increased the distance/losses by adding a 20km fiber spool with 4dB of loss. This was done also to increase the Raman noise from the classical co-propagating channels (i.e. a worst case situation for the transmission of quantum signals). In this combined configuration the key generation capability was above 20kbps, with coexistence of quantum and classical channels.
- Norte-Concepcion: $\sim$ 7dB losses, carrying only telecom channels (using the QKD network with a multi-hop/virtual link).

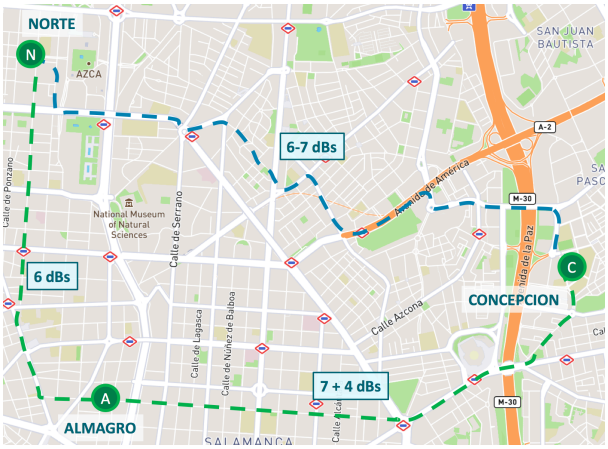

Fig. 1. The QKD testbed in Madrid: Map view.

In Fig. 1, we show a map of the network, also indicating the links with an approximation of the fibers path. The actual deployment might not follow this shortest path but a more complex and longer path due to external restrictions. Although there is no direct quantum link between Norte and Concepcion, the QKD keys provided by a SDN-based QKD network management (key management layer) over the trusted node Almagro in a multihop scheme are used here i.e. we have here a virtual QKD connection, as opposed to the direct, physical, connections through a direct and uninterrupted quantum channel.

The quantum channel is multiplexed together with other classical (data and management). The maximum number of co-propagated channels was 17, but this restriction came not from the QKD devices, but from the limited availability of the number of client ports in the optical networking equipment (standard Huawei's Optix OSN 1800 systems) used to multiplex/demultiplex the channels in the fibre.

III. Existing Proof-of-Transit Technique

The latest version of the PoT technique can be found in [12], which is an IETF working group document (WGD). In the next subsection we explain the general concept, before going into the details of the existing weaknesses of the scheme. The idea is to create a master secret which can be disclosed only if all participants in the scheme provide its share of the secret. In this sense, the verification node will prove if all nodes involved in a path (as show in Fig. 2) have provided its share to reconstruct the secret. The mathematical concept behind is the Shamir Secret Sharing (SSS) technique [18], which is based in the fact that a polynomial of degree $n-1$ (the secret) is uniquely determined when n interpolant points of the polynomial (the shares) are known.

A. Description

The steps described in the IETF's PoT WGD are as follows:

- The first step is to create the scheme. An entity (which could be a SDN controller) must create it. This includes choosing a big prime number, generating the two polynomials -$P_1(x)$ and $P_2(x)$- (for the given number of nodes in the path), selecting the points for each node and generating the (as defined in the WGD) Lagrange Polynomial Constants (LPCs) per point. Lagrange polynomial formalism is used throughout since once calculated the basis, that is fixed for a set of abscissae, the evaluation of the corresponding interpolation polynomial is fast. The first polynomial is the secret in the SSS scheme, while the second polynomial is used for randomising the scheme, so that it can be reutilised and not just for a single packet (for more information refer to [12]).
- After this calculation, the entity (controller) sends the computed PoT metadata (see Fig. 5 and related explanations for details) to each node in the path. This metadata will be stored by the node and used when required over the PoT flow.
- For any incoming packet, the source node in the path generates a random number between 0 and the prime p. This random value (RND) is used as the constant coefficient of the second polynomial. This procedure is done to randomise the scheme per packet, avoiding to regenerate the scheme for each new packet of the flow.
- Any node i (including the source and destination of the path) for a given packet j will perform the same operation in order to update the cumulative verification value (CML). At the first node, CML_0 is consider zero, as it is the first node generating the

cumulative value.

$$CML_i^j = (P_1(x_i) + (P_2(x_i) + RND^j)) * LPC_i + CML_{i-1} \pmod p$$

- The packets are sent between the nodes in the chain, using the encapsulation defined by the in-situ OAM datagrams [19]
- After updating the cumulative value with its own share, the destination/verificator node compares the cumulative value generated with the sum of the master secret plus the random number RND:

$$CML_n == Secret + RND \pmod p$$

- If the verification succeeds, the verification node removes the PoT header (metadata) from the packet and forwards it to the next node (outside the path).
- If the verification fails, an action will be taken with the packet based on predefined policies (e.g. drop the packet).

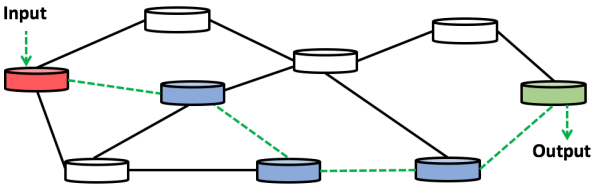

Fig. 2. Example of network and PoT path.

This technique allows to verify that a traffic has gone through a set of nodes within a path, but not the order within the path. For this reason, the PoT WGD also proposes using cumulative encryption technique, which will require the verification node to synchronize symmetric keys with every node in the path. Furthermore, the associated overhead per packet will be at least of the number of bits of the symmetric secret key (e.g. 256 bits if AES-256 is used), with the associated computational complexity (and hardware requirements) needed for the nested encryption and secret key creation (e.g. Diffie-Hellmann, which is computationally costly).

Section 8 of the draft claims that the security from the original SSS scheme (information theoretic secure - ITS) can be inherited and that, in consequence, it is impossible to break as long as some conditions are met: the polynomial must of degree k for chains of $k+1$ nodes and that the values used for the reconstruction are kept secret by the nodes. Also, the draft mentions the number of packets that can be validated by the scheme is similiar to the prime number chosen. In the next subsection we show that some of the assumptions are not valid, as a given attacker could potentially break the scheme and bypass a set of nodes by reading the OPoT packets metadata [19] (in a mixture of what the draft calls inter-node and inter-packets cryptanalisis). We evaluate the current weaknesses of the SSS-based scheme (not the nested encryption), to provide a better context for our solution.

B. Vulnerabilities

Initially we will assume that two packets $(packet^1, packet^2)$ can be captured just after the first node in the path. Let $P_1(x_i)$ be the result of the first polynomial for x_i (point assigned to $node_i$), and $P_2(x_i)$ be the result of the second polynomial without the constant part. Then let RND^j be the constant part of the second polynomial for a given $packet^j$ and CML_i^j be the cumulative value for a $packet^j$ after the $node_i$. LPC_i stands for Lagrange Polynomial Constant, meaning the Lagrange basis polynomial constant for the point x_i. Both x_i and LPC_i must be kept secret. Then, by capturing two packets we get:

$$\begin{aligned} CML_1^1 &= (P_1(x_1) + P_2(x_1) + RND^1) \\ &\quad * LPC_1 \pmod p \\ CML_1^2 &= (P_1(x_1) + P_2(x_1) + RND^2) \\ &\quad * LPC_1 \pmod p \end{aligned}$$

from where

$$\begin{aligned} &CML_1^1 - CML_1^2 \pmod p \\ &\quad = (RND^1 - RND^2) * LPC_1 \pmod p \end{aligned}$$

and

$$LPC_1 = \frac{CML_1^1 - CML_1^2}{RND^1 - RND^2} \pmod p$$

For a given node *i* the result will be:

$$\begin{aligned} CML_i^1 &= (P_1(x_i) + P_2(x_i) + RND^1) \\ &\quad * LPC_i + CML_{i-1}^1 \pmod p \\ CML_i^2 &= (P_1(x_i) + P_2(x_i) + RND^2) \\ &\quad * LPC_i + CML_{i-1}^2 \pmod p \end{aligned}$$

from where

$$\begin{aligned} &\big((CML_i^1 - CML_{i-1}^1) - \\ &(CML_i^2 - CML_{i-1}^2)\big) \pmod p \\ &= (RND^1 - RND^2) * LPC_i \pmod p \end{aligned}$$

$$LPC_i^{1,2} = \frac{(CML_i^1 - CML_{i-1}^1) - (CML_i^2 - CML_{i-1}^2)}{RND^1 - RND^2} \pmod p$$

Where we use $LPC_i^{1,2}$ to mean the LPC_i generated by the captured packets 1 and 2. The LPC_i value must be unique, but we still don't know the prime p used for the module operation, making its calculation hard. If l, $2 < l < 10!$ is the number of $LPC_i^{y,z} \in \mathbb{Z}$, then we could calculate $(l-1)!$ combinations of:

$$\begin{aligned} LPC_i^{y,z} &= LPC_i + m^{y,z} * p, m^{y,z} \in \mathbb{Z} \\ LPC_i^{v,w} &= LPC_i + m^{v,w} * p, m^{v,w} \in \mathbb{Z} \\ LPC_i^{y,z} - LPC_i^{v,w} &= (m^{y,z} - m^{v,w}) * p \end{aligned}$$

And then we can calculate (using the division results that belong to the set of integers) the set:

$\{(m^{1,2} - m^{1,3}) * p, ..., (m^{1,4} - m^{3,7}) * p, ...\}, (m^{k,l} - m^{r,s}) \in \mathbb{Z}$

From where we could try to derive the prime (e.g. $m.c.d$, decomposing several p and operating them to verify if they fit). If the prime p is obtained, the PoT schema is compromised and the rest of the demonstration is as follows.

CML_i^j and RND_i are public (openly encapsulated in the packet metadata). Therefore, assuming the traffic could be tampered at any point (and p has been obtained), we could calculate for a given setup of the PoT, the set $\{LPC_1, ..., LPC_{n-1}\}$ by tracing two different packets (and their PoT metadata).

Let's now assume that a third packet ($packet^3$) traverses the network. We will check if, by knowing RND^3, we could calculate CML_j^3 values. Although this cannot be done in principle, since the calculation of the random constant is done internally in the first node, we can start calculating CML_1^3:

$$\begin{aligned} CML_1^1 &= (P_1(x_1) + P_2(x_1) + RND^1) * LPC_1 \\ &\Rightarrow (P_1(x_1) + P_2(x_1)) * LPC_1 \\ &= CML_1^1 - (RND^1 * LPC_1) \end{aligned}$$

$$\begin{aligned} CML_1^3 &= (P_1(x_1) + P_2(x_1) + RND^3) * LPC_1 \\ &= (P_1(x_1) + P_2(x_1)) * LPC_1 + RND^3 * LPC_1 \\ &= CML_1^1 - (RND^1 * LPC_1) + RND^3 * LPC_1 \\ &= CML_1^1 + (RND^3 - RND^1) * LPC_1 \end{aligned}$$

For a given node *i* the result will be:

$$\begin{aligned} &(P_1(x_i) + P_2(x_i)) * LPC_i = \\ &CML_i^1 - CML_{i-1}^1 - (RND^1 * LPC_i) \end{aligned}$$

$$\begin{aligned} &CML_i^3 = \\ &= (P_1(x_i) + P_2(x_i) + RND^3) * LPC_i + CML_{i-1}^3 \\ &= (P_1(x_i) + P_2(x_i)) * LPC_i + RND^3 * LPC_i + \\ &\quad + CML_{i-1}^3 \\ &= CML_i^1 - CML_{i-1}^1 - (RND^1 * LPC_i) + \\ &\quad + RND^3 * LPC_i + CML_{i-1}^3 \end{aligned}$$

leading to:

$$\begin{aligned} CML_i^3 = CML_i^1 - CML_{i-1}^1 + \\ + (RND^3 - RND^1) * LPC_i + CML_{i-1}^3 \end{aligned} \quad (1)$$

Then, by tracing the traffic before and after a single network node *i*, we could bypass that node of the PoT path by getting the packet, and calculating the corresponding $CML_i^{new-packet}$. If the information is gathered at each hop of the path, we could also grow the secret up to the $n-1$ hop by getting the packet at the first node, bypassing the whole path. For example, using directly formula 1 on the second node produces:

$$\begin{aligned} CML_2^3 = CML_2^1 - CML_1^1 + \\ + (RND^3 - RND^1) * LPC_2 + CML_1^3 \end{aligned}$$

since all the values are know, CML_2^3 can be generated and then node 3 can be bypassed. This could be then grown for any node *i*. Note that these operations must be also done $\pmod p$. This assumes that the prime p can be disclosed as initially shown.

To verify that such an attack is feasible under normal conditions, where a low overhead is required, we generated different 32 bits prime numbers and their randomily generated PoT schemas, including the two polynomials, the points and the LPC_i for each node in the path. The simulation was implemented to capture the first 100.000 packets (out of a maximum of 2^{32} packets), failing if it did not find a valid solution (i.e. the prime number and the attack for a given packet). In most of them (above 85%) we could succesfully obtain the prime number and replicating the schema (gathering packets at any intermediate hop in the path). Further tests with smaller prime numbers were executed, as it is shown in the results section.

Incrementing the size of the prime number would increase the complexity of disclosing enough data to break the solution, but it will also increase the cost of growing the secret and, more importantly, will require to expand the PoT header for each packet, increasing substantially the overhead.

Also, [12] proposes an alternative solution to include order as a capability of the solution. This technique is based on nested encryption to generate a secret using a

symmetric ciphering method (e.g. AES). As it is defined, it is not just a complement to the existing proposal but more a new solution itself, as it does not need a SSS scheme to work in parallel. However, this also has some associated drawbacks: it is computationally more costly, it requires an additional overhead (at least the size of the key 256 of control information per packet) and it requires the node in charge of verification to securely generate symmetric keys with all the nodes in the path. For this last point, if keys are to be refreshed in a per packet basis, it will require the validation node to keep a continuous flow of secret keys with every node.

For this reason, we explored the possibility of integrating symmetric encryption algorithms not replacing, but working together with SSS scheme. This not just mitigates the security issues that the current solution contains, but can also be used to provide order to the PoT verification scheme.

IV. Proposed Enhancement

The purpose of this section is to provide a detailed description of the proposed solution extended to include order. Our technique enhances the solution in [12] while being compatible with it. It utilises standard techniques from traditional symmetric encryption algorithms and provides the flexibility to define multiple options/embodiments for providing different level of assurance (LoA). The next subsection provides a description and an example of our solution.

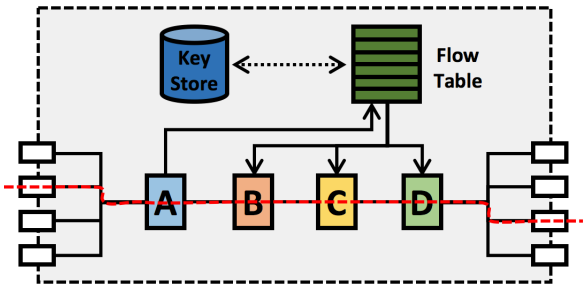

Fig. 3. High level view of the internal node steps for providing OPoT.

A. Description

To enhance the security of the existing solution, the proposed technique combines the SSS scheme with symmetric key encryption in a per-hop basis, securing the SSS metadata on each hop. This addition also allows to reduce the polynomials to only one, by keeping the random number generation at the source node of the path. Let us remark that our proposal is compatible with the existing one, as both polynomials can be kept (and also be exposed as a LoA capability). Note that the main difference of maintaining the two polynomials against using just one is to keep a constant coefficient (the secret) different from zero, so either possibility (two or one polynomials) is compatible with our proposal.

Fig. 3 shows a high level view of the internal logical components of a network device performing the OPoT. These components are divided in four steps (packet identification, decryption, SSS core, encryption). A detailed view of the steps for any intermediate node (i) in the path is as follows:

A The first step consists on the identification of the packet and verifies if it matches a routing/forwarding entry (in Fig. 3, from the flow table) that is associated with the OPoT mechanism. If so, it gathers the appropriate symmetric keys (previously exchanged/agreed with the previous nodes $i-1$ and $i+1$ within the path) from the key store and provides it to step B and step D for the decryption/encryption of the SSS metadata (CML and RND).

B The second step involves the disclosure of the ciphered SSS metadata within the packets header. When this process is finished and the metadata is updated (with the valid open values) it is handled to step C.

C During this step the node updates the SSS metadata by performing the very same reconstruction as described in the previous section using its own shares of the secret. This, as previously mentioned, can be done either for one:

$$CML_i = (f(x_i) + RND^j) * LPC_i + \\ + CML_{i-1} \pmod p$$

or two polynomials:

$$CML_i = (P_1(x_i) + (P_2(x_i) + RND^j)) * \\ * LPC_i + CML_{i-1} \pmod p$$

D The final step will require to cipher the data updated by the process in step C, using the appropriate symmetric keys (previously exchanged/agreed with the previous node $i+1$ within the path). After ciphering the SSS metadata, the node forwards the traffic to the network, using the information from the forwarding/routing table (as done in any traditional network).

It is important to note that the source and destination node slightly differ on some of the steps described above;

- Source node: In Step A there is no need for identifying a symmetric key, as traffic comes with no SSS header. In consequence, step B is not required. Step C will also require the source node to generate

a random number (between zero and the prime number) per packet. Step D works as defined before.
- Destination node: Step A and B works similarly as in the general description, but there is no need for gathering a next-node symmetric key (as it is the destination node). In consequence, step D is not required. Step C, apart from behaving as in the general description, it verifies the generated data (if the OPoT works) and removes the SSS metadata from the packet's header.

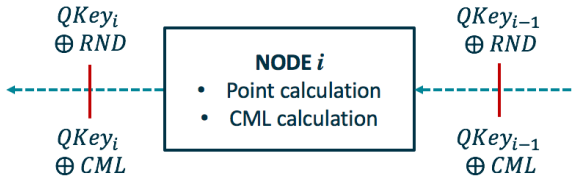

Fig. 4. High level view of the internal node components for providing OPoT.

Fig. 4 shows in a higher level view how the metadata from the SSS is treated before and after passing through the OPoT-enabled node. The proposed technique is compatible with novel network paradigms: a SDN controller could be the master entity in charge of creating the SSS data (prime, polynomials, points, $LPCs$) and deploying the OPoT path, while the solution is easily integrable and suits particularly well in virtualized environments (NFV). In addition, different key exchange techniques can be applied for providing keys at each points, depending on the available resources and the LoA associated with the traffic flow. For example, traditional key exchange algorithms (e.g. Diffie-Hellman), post-quantum algorithms or QKD could be used as the source of symmetric keys. Which one to use would depend on the available infrastructure (e.g. QKD modules, HSM encryptors), the LoA agreed for the service or the overhead acceptable for the traffic flow.

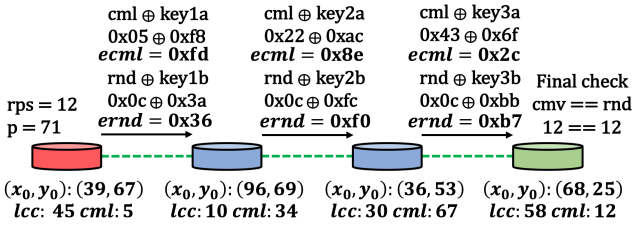

Fig. 5. Example of the scheme using a single polynomial combined with the per-hop encryption.

In Fig. 5 we show in detail a simple execution using four nodes (n_1, n_2, n_3, n_4) performing the proposed scheme. If a packet leaving the ingress node (n_1) goes to the third node (n_3), bypassing one intermediate node (n_2), the values used for the calculation will be a combination of cml and rnd, both XORed with the two intermediate (and different) keys used between the first and second nodes (n1, n2) and the second and third nodes (n2, n3). The data used in the example are:

Prime number: $p = 71$

Polynomial: $61x^3 + 55x^2 + 10x$ with no constant coefficient.

The nodes are given the values $\{xi, yi, lcci, p\}$

- $n_1 : \{39, 67, 45, 71\}$
- $n_2 : \{96, 69, 10, 71\}$
- $n_3 : \{36, 53, 30, 71\}$
- $n_4 : \{68, 25, 58, 71\}$

with no verification value, as the transmitted rnd_j will be used for verification, for a given packet j. In the figure, $ecml$ and $ernd$ stand respectively for encrypted cml and rnd values. The nodes show the internal steps involved in the OPoT for this specific path. The ingress node (n_1) generates the random value rnd to be used as the constant coefficient of the polynomial (also named rps, random polynomial secret), which is the secret. This random value, $(rnd = 12$ in the example), is transmitted with the packet and ciphered per hop. Each node in the path decrypts the OPoT metadata, reconstructs its part of the secret using its part of the share and the OPoT packets metadata, ciphers the new values and transmits them to the subsequent node in the path. The top part of Fig. 5 shows the encryption of the OPoT metadata, while the bottom part shows each nodes share and the generated cml. The verification node (n_4) finally checks if $cml == rnd$ (in this case, succeeding $12 == 12$), allowing the packet to be forwarded if the verification succeeds.

B. Key Improvements/Capabilities

Our proposed solution, while being compatible with the exiting proposal, brings some additional enhancements. Both the enhancements and the maintained capabilities are:

- The fundamental enhancement comes from the upgrade on the security of the existing scheme. The SSS metadata is ciphered per hop, avoiding that any evesdropper gathers and analyses the data to perform an attack similar to the one described above.
- A second (but no less important) enhancement is that our solution brings order to the solution. If the packet is not correctly forwarded between the nodes (e.g. it goes from the first to the third node), the node receiving the packet won't have the corresponding key, so it will wrongly decrypt the metadata, leading to a wrong construction of the secret.

- It is compatible with the existing solution, so the scheme can be randomised per packet.
- The first solution for reducing an attack is increasing the size of the prime number, therefore increasing the overhead. Our solution allows not to increase the size of the prime number and, in this sense, reduce the overhead of the solution.
- As mentioned above, our solution allows to simply use a single polynomial, while being compatible with the two polynomial approach of the existing solution.
- In the IETF WGD [12], the proposal for ordered PoT requires nested encryption. This solution needs some computational capabilities. It also needs that the final node has active key exchange systems with all the other nodes in the path and brings an overhead of at least the size of the key bits per packet. Our solution does not have such computational complexity (a simple XOR operation can be used) and the packet overhead is also reduced, as there is no need to increase the key size for increasing the security of the scheme.
- This solution can be complemented with different LoA profiles, based on multiple variables:
  - Size of the scheme (the size of the prime number): Increasing this value may affect the throughput, while keeping it small could be suitable for low rate services. Smaller schemes can be usable for more traffic, due to the improvements of our proposal.
  - Source of the symmetric keys: keys can be generated from different sources,

C. Future Possibilities

We note, without going into details, that there are approaches based on frequent key exchange, which are completely unrelated to the method presented in [12]. These are based on Message Authentication Code (MAC) algorithms [20] that ensure the integrity and the source of information of received messages. The said algorithms are known to be secure if an adequate frequency of exchange of secure key material is guaranteed. An extreme example of this class are MACs that ensure Information Theoretically Secure (ITS) message authentication [21] - ones that cannot be broken by any cryptoanalytic method, provided that the key is changed at every MAC application, they are absolutely secure, i.e. they preserve the secrecy of the message and are unforgeable by the adversary. The quality of the key material generated by QKD is a very good approximation to this requirement. It should be taken into account that present and foreseeable QKD key generation rates are insufficient for ITS MACs, if these would be applied to every single packet. However, packets can be aggregated in larger frames and ITS MACs can be applied to these, provided the nodes possess a source of ITS key generation. Naturally such a strategy comes at the expense of a loss of all packets in a frame (and the need of their resending) when a MAC fails. Alternatively, however, the application of non-ITS MACs can be used taking into account a careful calibration of the key exchange rate in order to meet a given security policy.

V. Experimental Results

A. Brute force attack on the existing solution

The brute force attack results are summarized in Figs. 6 and 7. In both graphs the x-axis shows the prime number used for the demonstration.

The first graph (Fig. 6) shows the average number of packets captured at each hop and used to break the scheme, calculating the $LPCs$ and the prime number. The second graph (Fig. 7) shows similar information but as the percentage of packets that are required to be listened to break the scheme for different prime numbers. The percentage is calculated by dividing the average number of packets by the total amount of packets assumed to be secured by the scheme (which is the prime number used to generate the scheme).

These tests were executed 10 times for each scheme, having 10 schemes for prime number amounting to a total of 100 tests for each prime number. For example, for the prime 1003161329, we run 100 tests with an avarage of 34890 packets captured to break the scheme (a minimum of 10058 and maximum of 71964 were observed). The graphs also indicate that, even when the required number of packets increases as we use bigger primes, the growth in the average value for each prime does not increase drastically (as fast as the prime number is increased) and, in this sense, the calculated percentage (average number of packet to break the system vs the prime number).

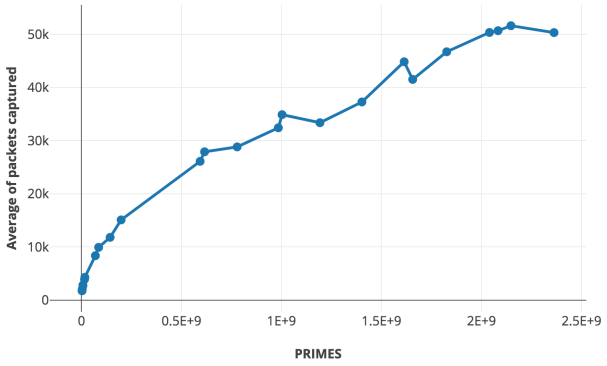

Fig. 6. Average of number of packets required to break a 5 nodes schema for a given prime number (x-axis).

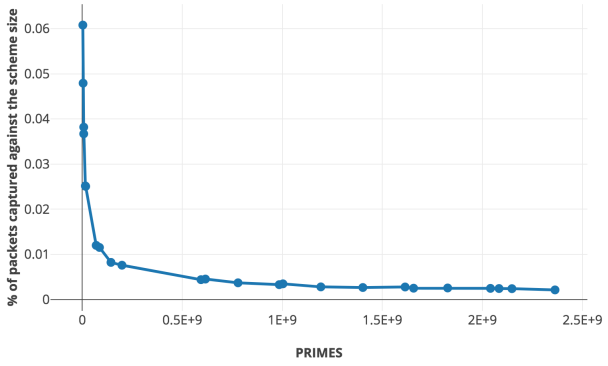

Fig. 7. Percentage of packets required to break a 5 nodes schema for a given prime number (average divided by the prime, which is assumed to be the maximum number of packets to be secured by the PoT).

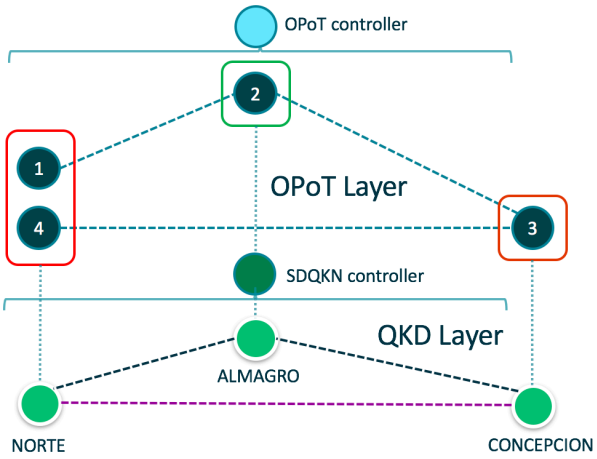

Fig. 8. Setup of the two logical layers within Madrid's Quantum network: the QKD layer (lower part) and the data, OPoT layer (upper part).

B. Demonstration over the Madrid QKD network

The experimental setup is composed by three PoP distributed in Madrid's production network, as described in the previous section. At each location, our physical testbed comprises:

- a QKD system (either a transmitter or receiver),
- a server that contains the SDN software for managing the QKD systems and the key stores, the software for the QKD system operation and the processes implementing the OPoT solution (the nodes that form the path and the controller), allocated in separate virtual machines.
- a Huawei's Optix OSN 1800 optical transmission system aggregating data, control and quantum channels. These are completely off-the-shelf devices.

Our two network layers (the data and QKD) are shown in Fig. 8. The lower layer shows the QKD domain. It is composed by the three physical QKD systems, its management software (internal, key stores and a SDN-agent) and a SDN controller, physically located within Almagro's PoP. The SDN agents at each location can communicate with the centralized controller via well-defined information models and interfaces (RESTconf). Through them, the controller is capable of deploying and switching the physical QKD channels (Almagro-Norte and Almagro-Concepcion), creating a virtual QKD link using Almagro's PoP as a secure key relay node, and detecting, tracking and handling the incoming key-consuming applications (in our case, the four nodes performing the OPoT).

The upper layer (data) is composed by four nodes. The path connecting the nodes starts in a process located in Norte's PoP. It transmits the data to the second node/process, located in Almagro. From Almagro the path goes to the process at Concepcion, which finally closes the loop by sending back the data to a second process located in Norte. In this way, we make sure that the applications consume keys from every (either physical or virtual) QKD link. The implemented application, selected only as a simple showcase, was the transmission of raw messages simulating a chat within a secure locations (the PoPs). Other, more significant use cases, for example, the verification of a QKD key relay (or virtual link) over a QKD network have been also implemented.

The workflow for this demonstration is as follows:

- Initially, our source node needs to communicate with the controller for requesting the deployment of an OPoT flow/path. This request (as other messages in the workflow) has been implemented over UDP datagrams. A more network-oriented approach would be to implement the channel following SDN standards (e.g. OpenFlow/NETCONF), while for a quantum-safe communication between node and controller a solution like [5] would be better suited.
- Upon request reception, the controller computes the path (in our case, with no constrains from the source node), selects a prime number, generates the two polynomials/points/LPCs for the SSS reconstruction and sends these values to every node in the path. Note that we use the two polynomials scheme to follow closely the IETF WGD proposal although, as it was explained before, this is not mandatory. For a given test, the data generated (randomly) by the node was:
  First polynomial:
  $3293992791x^3 + 3749031361x^2 + 2496180873x + 1507514772$
  Second polynomial:
  $3950162548x^3 + 1479635150x^2 + 3577848592x$
  Data to $node_1$:

$\{App : 10, P_1(x_1) : 636367073,$
$P_2(x_1) : 2174438441, LPC_1 : 3959207178,$
$prime : 4111484371, next : 172.16.0.100\ 5445,$
$enc_{next} : true\}$
Data to $node_2$:
$\{App : 10, P_1(x_2) : 560148219,$
$P_2(x_2) : 2288072367, LPC_2 : 316268028,$
$prime : 4111484371, prev : 172.16.20.100 : 5445,$
$enc_{prev} : true, next : 172.16.40.100 : 5445,$
$enc_{next} : True\}$
Data to $node_3$:
$\{App : 10, P_1(x_3) : 360017364,$
$P_2(x_3) : 3757300196, LPC_3 : 1065940351,$
$prime : 4111484371, prev : 172.16.0.100 : 5445,$
$enc_{prev} : true, next : 172.16.20.100 : 5446,$
$enc_{next} : true\}$
Data to $node_4$:
$\{App : 10, P_1(x_4) : 2088721923,$
$P_2(x_4) : 750885585, LPC_4 : 2881553186,$
$prime : 4111484371, prev : 172.16.40.100 : 5445,$
$enc_{prev} : true, secret : 1507514772\}$
where App identifies the specific application/flow, $P_i(x_j)$ is the calculated point for node j using the polynomial i, LPC_j is the Lagrange polynomial constant for node j, $prime$ is the prime number chosen by the controller for the solution, $prev$ and $next$ are the previous and next nodes in the path, enc_x defines if the previous or next nodes require encryption for the SSS metadata and $secret$ is the constant coefficient of the first polynomial, following the standard IETF scheme and the SSS algorithm.

- When the nodes receive the OPoT parameters, if encryption is active, the node extracts the keys from the QKD domain. At each request, the SDN controller is informed by the local agents, so then the controller knows which applications are connected where and can monitor and allocate resources (configuring each agent to reserve a pool of keys). In Fig. 9 we show the flow of keys for each link used to secure the OPoT metadata during this test.
- At this point, all the necessary configuration has been sent to the nodes, so the OPoT scheme can be used by the nodes.
- The source node, using the App value provided by the controller (implemented for this specific application), is able to send a message to the destination node through the path which has been already configured. This first message transmitted, shown in Fig. 10, is *"hello, my friend"*. The capture also shows the OPoT encrypted metadata, i.e. $ernd$ and $ecml$.
- Fig.11 shows how the metadata is modified at each node in order to cipher, modify and decrypt the values to grow the secret, following the steps described in the previous sections.

	Key Stream ID	Key Stream
Prev Node 1 Next	N/A	N/A
	eea4b659-7035-0898-35ec-a4f64ef8b54a	fa85c6ffe4ce0f7d59f06f0b4a38...
Prev Node 2 Next	eea4b659-7035-0898-35ec-a4f64ef8b54a	fa85c6ffe4ce0f7d59f06f0b4a38...
	9c7f687b-3a44-bfca-ecc4-020a24a51dbe	e6e3d5b5591e0d662d413452a70...
Prev Node 3 Next	9c7f687b-3a44-bfca-ecc4-020a24a51dbe	e6e3d5b5591e0d662d413452a70...
	cd954643-3434-3c11-5329-b516237fc6db	93d692de1f0f6768ab095fd047fbc...
Prev Node 4 Next	cd954643-3434-3c11-5329-b516237fc6db	93d692de1f0f6768ab095fd047fbc...
	N/A	N/A

Fig. 9. QKD keys used to secure the OPoT metadata for each hop.

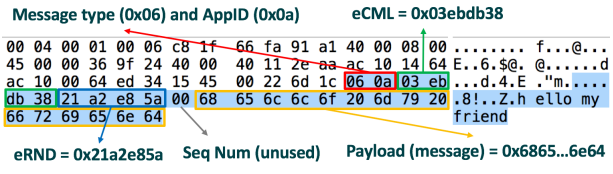

Fig. 10. OPoT packet captured after leaving the first node.

	@ Node 1	@ Node 2	@ Node 3	@ Node 4
Enc. RND (recv)	NA	0x21a2e85a	0x3dc4fb10	0x48f1bc7b
Dec. RND Key	NA	0xfa85c6ff	0xe6e3d5b5	0x93d692de
Clear RND	0xdb272ea5	0xdb272ea5	0xdb272ea5	0xdb272ea5
Enc. RND Key	0xfa85c6ff	0xe6e3d5b5	0x93d692de	NA
Enc. RND (send)	0x21a2e85a	0x3dc4fb10	0x48f1bc7b	NA
Enc. CML (recv)	NA	0x03ebdb38	0xf59796d6	0x34e7a23d
Dec. CML Key	NA	0xe4ce0f7d	0x591e0d66	0x1f0f6768
Clear CML	NA	0xe725d445	0xac899bb0	0x2be8c555
Updated CML	0xe725d445	0xac899bb0	0x2be8c555	0x3FF1C266
Enc. CML Key	0xe4ce0f7d	0x591e0d66	0x1f0f6768	NA
Enc. CML (send)	0x03ebdb38	0xf59796d6	0x34e7a23d	NA

Final verification → CML value :: 0x3FF1C266 == 0x3FF1C266 :: Verification value

Fig. 11. Table exposing the keys, CML and RND values at each node within the path for a given execution.

In the last stage, the verificator/destination node also updates the CML value and finally checks if the calculated (grown) value is the same as the secret plus the random number for a given packet i.

$$CML_n^i = RND^i + Secret$$

In the implementation, a packet being non-compliant with the scheme (i.e. not following the order, being modified at some point or skipping one or multiple nodes)

was directly dropped by the verificator. Nonetheless, this solution is also suitable with traditional networking schemes and policies, so other actions could be also taken if the scheme fails on verifying the path.

The enhanced version shall not be very demanding in terms of secret key updates. If the RND values are not repeated for the same scheme and the random number generation does not imply any additional security breach, the secret masks can be reutilized by the network elements. In this sense, considering also the data provided by section 4 in the draft, in a 32 bit scheme with 100 Gbps traffic chain, a scheme (set of polynomials) will last for 22 seconds. Updating the masks every second will mean 22 updates per a given PoT scheme, meaning a secret key consumption of 64 bits per second (which is negligible, based on the rates that can be provided by modern QKD systems). The most demanding resource would be the random number generation which, as an example, for a given 10Gbps traffic service, assuming a MTU of 1500, we will approximately have a total of 840.000 packets per second. This implies a maximum generation of 27Mbps, which is multiplied by 10 for 100 Gbps services. Other quantum services, as quantum random number generators [22], can be the source to cope with this demand, being installed in commodity data centers.

VI. Future Work

The next steps for this research can be divided in two. The first would be from the standardization side, where the OPoT enhancement is being proposed in the newest version of the IETF draft. This may require further work in terms of datagram (e.g. dynamic rekeying synchronization for a given scheme using iOAM headers) and control plane definitions (extensions on the current version of the YANG models). The second will imply further research on the security implications of the PoT definition. This work is mainly focused on studying the original assumptions of the PoT proposal, to demonstrate that the scheme could be bypassed using far less packets that initially stated. The attack is an illustrative example to show that the scheme is not applicable for the amount of data stated in the document. Additional performance enhancements can be applied to the attack, as some parts are currently based on a ”brute force” approach (validating one to all), whilst some searchs and validation could be parallelized.

In addition, the principles on which the MQN have been developed, fundamentally related to Software-Defined Networking, are being defined and standardized within the European Telecommunications Standards Institute (ETSI), under the Industry Specification Group on QKD. The model defined for abstraction and management of QKD resources will require several incremental iteractions to include further use cases and broader parametrization, as the QKD technology itself is developed and standardized.

VII. Conclusions

Quantum key distribution networks are the key measure for implementing quantum-safe communications. This work presents a security use case that is implemented on top of the recently deployed Madrid Quantum Network. In the context of virtualized services and next generation protocols, new techniques for verifying the traffic path/flow are necessary for security and other reasons (e.g. legal interception). This work analyzes the IETF's proposal for a proof-of-transit technique based on Shamir's Secret Sharing and its associates vulnerabilities. We find that the security is lower than expected when using the standard method, and propose an enhancement that avoids the associated security threats while being compatible. The new scheme also provides order, an important property to secure the network. This improvement, based on symmetric encryption, is also demonstrated using a state-of-the-art technique, QKD, for providing secret keys and integrate them in the system. The use over a non-standard infrastructure highlights the flexibility of the scheme and the fact that it can be future-proofed even in the case of a major computing breaktrough, like the availability of a quantum computer. The findings presented in this work have been presented at the IETF and have been promoted as the preferred solution for ordered proof-of-transit [23]. The presented technique is implemented over the first QKD network deployed in a production environment and following SDN principles.

Acknowledgment

The authors would like to thank the Spanish Ministry of Economy and Competitiveness, for the grant CVQuCo, TEC2015-70406-R, FEDER-MINECO, the Madrid's regional government, Comunidad Autonoma de Madrid, for the project Quantum Information Technologies Madrid, QUITEMAD+ S2013/ICE-2801, the FET Flagship on Quantum Technologies, European Unions Horizon 2020 research and innovation programme under grant agreement No 820466: Continuous Variable Quantum Communications (CiViQ) and the team of Transport and IP Connectivity in Telefónica Spain for their support to this activity.

References

[1] E. Diamanti, H.-K. Lo, B. Qi, and Z. Yuan, "Practical challenges in quantum key distribution," *Npj Quantum Information*, vol. 2, p. 16025, 11 2016. [Online]. Available: http://dx.doi.org/10.1038/npjqi.2016.25

[2] V. Martin, J. Martinez-Mateo, and M. Peev, "Introduction to quantum key distribution," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, 2017, pp. 1–17. [Online]. Available: https://doi.org/10.1002/047134608X.W8354

[3] V. Martin, A. Aguado, P. Salas, A. Sanz, J. Brito, D. R. Lopez, V. Lopez, A. Pastor, J. Folgueira, H. H. Brunner, S. Bettelli, F. Fung, L. C. Comandar, D. Wang, A. Poppe, and M. Peev, "The madrid quantum network: A quantum-classical integrated infrastructure," in *OSA Advanced Photonics Congress (AP) 2019 (IPR, Networks, NOMA, SPPCom, PVLED)*. Optical Society of America, 2019, p. QtW3E.5. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=Networks-2019-QtW3E.5

[4] A. Aguado, V. Lopez, D. Lopez, M. Peev, A. Poppe, A. Pastor, J. Folgueira, and V. Martin, "The engineering of software-defined quantum key distribution networks," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 20–26, July 2019.

[5] A. Aguado, V. Lopez, J. Martinez-Mateo, T. Szyrkowiec, A. Autenrieth, M. Peev, D. Lopez, and V. Martin, "Hybrid conventional and quantum security for software defined and virtualized networks," *J. Opt. Commun. Netw.*, vol. 9, no. 10, pp. 819–825, Oct 2017.

[6] A. Aguado, V. Lopez, J. Martinez-Mateo, M. Peev, D. Lopez, and V. Martin, "Vpn service provisioning via virtual router deployment and quantum key distribution," in *Proc. Optical Fiber Conference (OFC)*, 2018.

[7] ——, "Virtual network function deployment and service automation to provide end-to-end quantum encryption," *J. Opt. Commun. Netw.*, vol. 10, no. 4, pp. 421–430, Apr 2018. [Online]. Available: http://jocn.osa.org/abstract.cfm?URI=jocn-10-4-421

[8] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment routing architecture," RFC 8402, 2018.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, March 2008.

[10] "Network functions virtualisation (nfv); architectural framework," in *ETSI GS NFV 002 V1.2.1*, 2014-12.

[11] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture," RFC 7665, 2015.

[12] F. Brockners, S. Bhandari, S. Dara, C. Pignataro, J. Leddy, S. Youell, D. Mozes, and T. Mizrahi, "Proof of transit," draft-ietf-sfc-proof-of-transit-00, 2018.

[13] M. Martinello, M. Ribeiro, R. de Oliveira, and R. de Angelis Vitoi, "Keyflow: a prototype for evolving SDN toward core network fabrics," *Network, IEEE*, vol. 28, no. 2, pp. 12–19, March 2014.

[14] F. Rubina, "One-time pad cryptography," Cryptologia, 1996.

[15] F. Laudenbach, C. Pacher, C. F. Fung, A. Poppe, M. Peev, B. Schrenk, M. Hentschel, P. Walther, and H. Hbel, "Continuous-variable quantum key distribution with gaussian modulation – the theory of practical implementations," *Advanced Quantum Technologies*, 2018.

[16] H. H. Brunner, L. C. Comandar, F. Karinou, S. Bettelli, D. Hillerkuss, F. Fung, D. Wang, S. Mikroulis, Q. Yi, M. Kuschnerov, A. Poppe, C. Xie, and M. Peev, "A low-complexity heterodyne cv-qkd architecture," in *2017 19th International Conference on Transparent Optical Networks (ICTON)*, July 2017, pp. 1–4.

[17] F. Karinou, H. H. Brunner, C. F. Fung, L. C. Comandar, S. Bettelli, D. Hillerkuss, M. Kuschnerov, S. Mikroulis, D. Wang, C. Xie, M. Peev, and A. Poppe, "Toward the integration of cv quantum key distribution in deployed optical networks," *IEEE Photonics Technology Letters*, vol. 30, no. 7, pp. 650–653, April 2018.

[18] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, November 1979.

[19] F. Brockners, S. Bhandari, C. Pignataro, H. Gredler, J. Leddy, S. Youell, T. Mizrahi, D. M. anf P. Lapukhov, R. Chang, D. Bernier, and J. Lemon, "Data fields for in-situ oam," draft-ietf-ippm-ioam-data-05, 2019.

[20] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[21] M. N. Wegman and J. L. Carter, *J. Comput. Syst. Sci.*, vol. 22, p. 265, 1981.

[22] M. Herrero-Collantes and J. C. Garcia-Escartin, "Quantum random number generators," *Rev. Mod. Phys.*, vol. 89, p. 015004, Feb 2017. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.89.015004

[23] F. Brockners, S. Bhandari, S. Dara, C. Pignataro, J. Leddy, S. Youell, D. Mozes, T. Mizrahi, A. Aguado, and D. Lopez, "Proof of transit," draft-ietf-sfc-proof-of-transit-02, 2019.